Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Entorno personalizado de realidad virtual para la rehabilitación de personas con diversidad funcional

Estudante: Manuel Lagos Rodríguez
Dirección: Javier Pereira Loureiro
Ángel Gómez García

A Coruña, novembro de 2020.

A mis padres, por darme los medios para llegar hasta aquí

Agradecimientos

A mi familia, amigos, compañeros de clase y a todo aquel que, en algún momento, me dedico unas palabras de apoyo.

A Javier y a Ángel, por permitirme realizar este proyecto, por vuestra ayuda y por vuestra cercanía.

A Cárlos Pérez, de Cogami, por su colaboración en la parte de modelado 3D.

A Thais Pousada, por orientarme en lo relativo a terapias a seguir con personas con diversidad funcional.

A cada uno de los profesores que me acompañaron durante estos cuatro años y a todo el personal de la FIC, predispuestos a ayudar en todo momento.

Resumen

La realidad virtual permite generar un entorno de gran realismo, a la vez que logra la inmersión del usuario en el mismo. La finalidad de este proyecto, es emplear dicha tecnología como herramienta complementaria en terapias con personas con diversidad funcional física y/o cognitiva.

A través de un software específico se han desarrollado cuatro escenarios que representan entornos y situaciones de nuestra vida cotidiana. Tres de los escenarios están orientados a tratar problemas de movilidad en las manos como, por ejemplo, el agarre de objetos o el movimiento de muñeca. El escenario restante está destinado a mejorar problemas de índole cognitivo, en concreto, la identificación de elementos para realizar una tarea común de nuestro día a día.

Durante la ejecución de un entorno, el usuario tendrá conocimiento sobre su progreso en la procura de los objetivos a cumplir mediante señales visuales y/o sonoras. Tras finalizar la ejecución, se generará un informe con los datos de seguimiento de las manos que, junto a la visualización en directo de la prueba, permitirá al terapeuta valorar la evolución del usuario.

Abstract

Virtual reality allows to generate an environment of great realism, while achieving the immersion of the user in it. The purpose of this project is to use this technology as a complementary tool in therapies with people with physical and/or cognitive functional diversity.

Through specific software, four scenarios have been developed that represent environments and situations of our daily life. Three of the scenarios are aimed at treating mobility problems in the hands, such as grasping objects or wrist movement. The remaining scenario is intended to improve cognitive problems, specifically, the identification of elements to perform a common task of our day to day.

During the execution of an environment, the user will have knowledge about her progress in the pursuit of the objectives to be fulfilled by means of visual and/or sound signals. After completing the execution, a report will be generated with the monitoring data of the hands that, together with the live visualization of the test, will allow the therapist to assess the evolution of the user.

Palabras clave:

- Realidad virtual
- Entorno inmersivo
- Diversidad funcional cognitiva
- Movilidad reducida
- Unity
- HTC VIVE
- Leap Motion

Keywords:

- Virtual Reality
- Immersive environment
- Cognitive functional diversity
- Reduced mobility
- Unity
- HTC VIVE
- Leap Motion

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Habitualmente, en las terapias realizadas con personas con diversidad funcional, se emplean una serie de elementos que configuran un entorno apropiado para tratar las dificultades del usuario. Para la configuración de estos escenarios, es posible encontrarnos con limitaciones físicas o económicas. Físicas, en cuanto a las características del edificio en el que se imparten las terapias. Económicas, en cuanto al desembolso a realizar en la adquisición de diversos materiales, que se podrían usar para modelar los diferentes entornos. Un ejemplo, lo podemos encontrar en el hospital *Parc Sanitari Pere Virgili* de Barcelona, en donde cuentan con una sala de terapia ocupacional que simula un piso real, disponiendo de estancias como una cocina o un baño [1].

Podemos definir la realidad virtual como *aquel tipo de representaciones generadas digitalmente que pretenden producir el mismo tipo de efectos perceptivos que los objetos sensibles de la realidad física de la vida cotidiana y que reacciona ante la acción del hombre de forma semejante a como lo hace esa realidad* [2]

Analizando los conceptos anteriores, podemos plantearnos la siguiente cuestión, ¿por qué no aprovechar el potencial de la realidad virtual en las terapias con personas con diversidad funcional?

Precisamente, este será el motivo que justifique el desarrollo del presente proyecto. La realidad virtual nos permite representar escenarios totalmente configurables y de gran realismo, disminuyendo considerablemente las limitaciones físicas. Así, podemos programar entornos personalizados, añadiendo cualquier elemento que se nos ocurra a través del modelado 3D.

Fundamentalmente, sería necesario un equipo de realidad virtual y un grupo de desarrollo cuyo cometido será el programar los entornos. Es fácil pensar en que, el mantenimiento de un equipo humano que desarrolle los diferentes escenarios, pueda suponer un gran coste para la entidad encargada de las terapias. No obstante, todo dependerá, por una parte, de la dimensión del entorno a utilizar para una determinada actividad y del recorrido que pueda tener la

misma. Así, no es lo mismo crear un entorno simulando la cocina completa de un hogar, a un ejercicio en el que se manipule un elemento concreto, como un cubierto. Como tampoco es lo mismo, desarrollar un escenario muy concreto para trabajar una dificultad específica en una persona a un entorno mucho más completo, que se pueda utilizar frecuentemente a lo largo del tiempo.

1.1 Motivación

La primera causa que motivo la realización de este proyecto, es profundizar en una tecnología que, a pesar de estar extendida en diversos ámbitos, sigue siendo desconocida para muchas personas.

Por otra parte, el crear un entorno de realidad virtual implica una parte de diseño gráfico y una parte de programación que dote de funcionalidad a los diferentes modelos, y precisamente, esta última parte se convierte en otra de las motivaciones, ya que permite aplicar en gran medida los conocimientos adquiridos en el grado cursado.

Por último, una causa que tuvo un gran peso, es el hecho de que el producto final podrá ser empleado por entidades sin ánimo de lucro, que trabajan en mejorar las dificultades presentes en personas con diversidad funcional. Es una forma de complementar las herramientas ya utilizadas por los profesionales de estas entidades, a la vez que se presenta una tecnología tan versátil como es la realidad virtual. Además, aunque no era una de las motivaciones iniciales, a causa de los tiempos que nos está tocando vivir a raíz del virus *COVID-19*, el hecho de no ser imprescindible interactuar con objetos físicos, se ha convertido en un aliciente más, ya que contribuye a reducir el riesgo de propagación por contacto durante el desarrollo de terapias que impliquen la manipulación de objetos.

1.2 Objetivos

La misión principal de este proyecto consiste en diseñar e implementar una herramienta de realidad virtual compuesta por una serie de entornos que simulen situaciones de la vida cotidiana, con la finalidad de ayudar al personal especializado en las tareas de rehabilitación de personas con diversidad funcional física y/o cognitiva.

Los entornos contarán con diversos elementos (objetos 3D), con los que el usuario podrá interactuar. Además, cada escenario se modelará en forma de prueba, con una serie de objetivos que el usuario deberá conseguir. De este modo, se pretende que la persona que ejecuta la prueba pueda entrenar y mejorar progresivamente sus dificultades, a la vez que el terapeuta responsable, pueda evaluar dicho progreso supervisando en directo la ejecución de la misma o consultando a posteriori un informe de actividad generado por la herramienta.

En concreto, se crearán 4 escenarios, tres destinados a tratar problemas de movilidad en las manos y uno orientado a problemas de índole cognitivo. En cada uno de ellos será necesario definir la escena y cargar los modelos 3D que la componen, implementar las acciones que permitan su manipulación y los flujos para alcanzar los objetivos propuestos.

A continuación, se indican los entornos a crear y su funcionalidad:

- **Escenario 1:** Contará con un grifo cuya apertura se acciona con el movimiento rotacional de una llave. El usuario deberá abrir el grifo, llenar una jarra de agua y cerrar el mismo. Con ello se persigue mejorar dificultades presentes en el movimiento de muñeca. Para desarrollar este escenario, se importarán los modelos 3D de los objetos con los que se interactuará y se crearán una serie de scripts que permitan accionar dicho grifo, generar visualmente un flujo de agua, proporcionar *feedback* al usuario y exportar un informe de actividad.

- **Escenario 2:** Implica el manejo de cubiertos, en concreto, el de un cuchillo y el de un tenedor. El objetivo será simular el corte de un filete. Se pretende mejorar el agarre de dichos cubiertos y los movimientos de las manos para llevar a cabo la acción de cortar. Para configurar dicho entorno, se importarán los modelos 3D de los diferentes objetos y se programarán las funcionalidades que permitan simular los cortes en el filete, proporcionar datos de progreso al usuario y exportar un informe de la prueba realizada.

- **Escenario 3:** Su objetivo es mejorar el agarre de elementos. Para ello se contará con un frutero y varias piezas de fruta. El usuario deberá colocar dichas piezas dentro del frutero. El desarrollo del escenario implicará, por una parte, importar los diseños de cada una de las piezas de fruta y del frutero. Por otra parte, se programará un conjunto de scripts que permitan detectar que frutas se han introducido en el frutero, ofrecer *feedback* al usuario y generar un informe una vez concluida la prueba.

- **Escenario 4:** En este entorno, se tratará una dificultad relacionada con la identificación de elementos, un problema de índole cognitiva. Para ello, habrá que seleccionar entre varios objetos, los necesarios para preparar el desayuno y colocarlos en una mesa. Al igual que en el resto de entornos, la fase de desarrollo comenzará por importar los modelos 3D y continuará con la programación de las funciones necesarias para detectar que se están seleccionando los elementos correctos para preparar el desayuno, indicar al usuario datos de progreso en la actividad y exportar un informe una vez finalizada la prueba.

1.3 Estructura de la memoria

A continuación, se muestra la organización de esta memoria junto a una breve explicación de lo que comprende cada capítulo:

- **Capítulo 1 - Introducción:** Ofrece una contextualización al lector, indica la motivación del trabajo y los objetivos marcados en el mismo.
- **Capítulo 2 - Estado del arte:** Expone el resultado de la búsqueda y análisis de herramientas similares en la actualidad.
- **Capítulo 3 - Fundamentos tecnológicos:** Describe cada uno de los recursos utilizados en el desarrollo del proyecto.
- **Capítulo 4 - Metodología:** Define la metodología utilizada para el desarrollo y su adaptación al proyecto.
- **Capítulo 5 - Planificación y estimación de costes:** Expone las diferentes fases de elaboración y el cálculo de costes del proyecto.
- **Capítulo 6 - Desarrollo:** Describe el proceso de desarrollo del proyecto. Se detalla cada una de las iteraciones en las que se dividió finalmente el trabajo.
- **Capítulo 7 - Conclusiones:** Analiza los resultados obtenidos tras finalizar el proyecto.
- **Capítulo 8 - Líneas futuras:** Expone posibles ampliaciones del proyecto.

Capítulo 2

Estado del arte

En este capítulo se muestra el resultado de analizar diferentes estudios o herramientas de realidad virtual, con un propósito similar a la propuesta a desarrollar en este trabajo. Se expondrá una idea general de cada uno, destacando aquellos puntos que sean de interés a afectos de comparación con el presente proyecto.

2.1 Realidad Virtual Inmersiva en personas mayores: estudio de casos

En el artículo titulado ”Realidad Virtual Inmersiva en personas mayores: estudio de casos” [3], desarrollado por la Universidad de Vigo, se realiza un análisis del uso de realidad virtual con cuatro personas mayores, dos de ellos parkinsonianos. Aunque el tipo de usuario no es el mismo que el de este proyecto, si lo son ciertos objetivos que tratan en el estudio, como el agarre de objetos. En este análisis, en cuanto a la parte hardware emplean las gafas de realidad virtual *HTC VIVE*, dos mandos para las manos y los sensores que delimitan el espacio de juego. El equipo es igual al usado en este TFG, salvo que a mayores, se cuenta con un dispositivo denominado *Leap Motion* [4], que reconoce nuestras manos y nos permite interactuar directamente con ellas, evitando el uso de mandos.

En cuanto a la parte software, en este estudio se utilizan los siguientes entornos de juego:

- **TheBlu**: Es una aplicación disponible en la biblioteca *Viveport*. Consiste en una simulación del fondo marino, a la vez que permite al usuario interactuar tocando elementos.

- **NVIDIA VR FunHouse**: Es un juego disponible en la tienda de *Steam VR*. Recrea una feria, permitiendo la inmersión del usuario y la interacción con diferentes elementos,

mediante agarres o lanzamientos, por ejemplo. En la Figura 2.2 , podemos observar uno de los escenarios de dicho juego.

Figura 2.1: Escenario del juego *NVIDIA VR FunHouse*

Como resultado, se indica que todos los participantes finalizaron las pruebas correctamente y que la experiencia fue valorada muy positivamente por parte de los mismos.

Destacan que una de las personas usaba silla de ruedas y que, además, a causa de su patología, contaba con cifosis dorsal, lo que le dificultaba la visualización total del entorno virtual. Para solventar este inconveniente, inclinaron su silla durante la ejecución de la actividad.

2.2 Realidad virtual para la rehabilitación de personas que han sufrido un ictus

Se trata de un proyecto [5] realizado por investigadores de la Universidad Politécnica de Barcelona en colaboración con la Asociación Diversidad Funcional de Osona (ADFO). En dicho proyecto, se han desarrollado dos aplicaciones de realidad virtual, destinadas a tratar problemas de diversidad funcional física y cognitiva respectivamente, y orientadas principalmente a personas que han sufrido un ictus.

En cuanto a la parte hardware observamos que, nuevamente, se usan unas gafas de realidad virtual *HTC VIVE*, dos mandos y los sensores que delimitan el espacio de juego. Respecto al software, los entornos están desarrollados por los investigadores del proyecto y, por lo que podemos observar en una de las imágenes, han sido creados mediante la herramienta *Unity*.

El escenario destinado a la rehabilitación física (Figura 2.2), contiene un personaje virtual que guía al usuario en la realización de diferentes movimientos. Los movimientos son captados por los sensores del equipo, permitiendo comprobar si el usuario realiza las acciones indicadas

por el personaje virtual. Además, los resultados se almacenan en una base de datos para su posterior valoración por un fisioterapeuta.

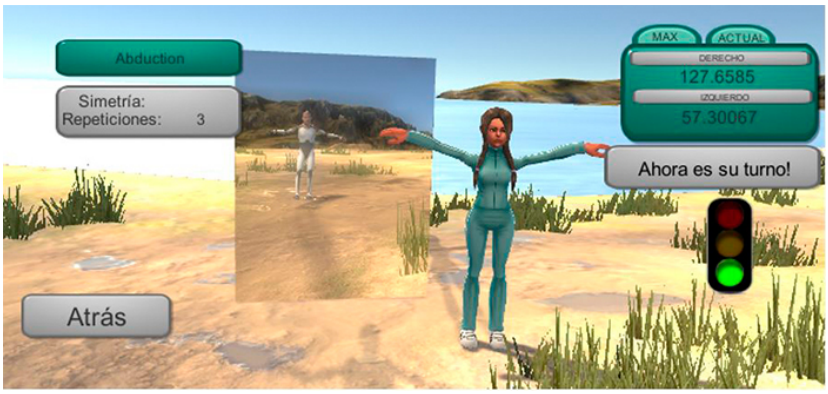

Figura 2.2: Escenario de rehabilitación física

En lo relativo al tratamiento de problemas de índole cognitivo, desarrollan un entorno inmersivo que simula la acción de comprar en un supermercado. El objetivo es que los usuarios trabajen capacidades como la memoria, la coordinación o la atención a partir de una tarea a realizar en nuestro día a día.

El planteamiento del proyecto es muy similar al del presente TFG. Además de emplear el mismo equipo de realidad virtual, los escenarios también se desarrollaron de forma personalizada, orientándolos al tratamiento de una serie de problemas de diversidad funcional muy concretos.

2.3 WalkinVR

WalkinVR [6] es un programa que tiene por objetivo hacer accesible la realidad virtual a personas con diversidad funcional. Adapta cualquier juego o entorno de realidad virtual del catálogo de *Steam VR*. Además, es compatible con múltiples equipos VR, como *HTC VIVE* o *Oculus Rift*.

En consecuencia, esta aplicación presenta un gran atractivo en el contexto de nuestro trabajo ya que, además de permitir el uso de entornos de realidad virtual por puro divertimento, posibilita el usar diversos juegos como medio de rehabilitación en personas con diversidad funcional.

A continuación, presentamos las funciones principales del programa:

- **Movimiento y rotación virtual**

  Función orientada a usuarios con problemas de movilidad como, personas que usen sillas de ruedas o que estén encamadas. Como podemos observar en la Figura 2.3, esta

opción nos permite realizar los diferentes movimientos del juego mediante los mandos del equipo de realidad virtual.

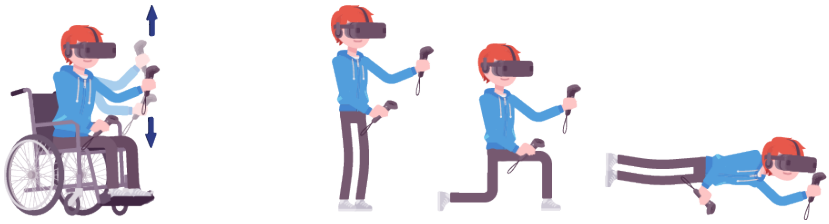

Figura 2.3: *WalkinVR* - Movimiento y rotación virtual

- **Jugabilidad con asistencia**

  Función pensada para personas con algún trastorno de movilidad, que limite significativamente su interacción con los controladores para desplazarse por el juego. Tal y como podemos observar en la Figura 2.4, si seleccionamos esta opción, otra persona podrá realizar los diferentes movimientos mediante un mando de *Xbox*.

Figura 2.4: *WalkinVR* - Jugabilidad con asistencia

- **Ajuste de posición del controlador**

  Esta opción está destinada a personas con trastornos como, por ejemplo, la atrofia muscular espinal, distrofia muscular o tetraplejia. Permite solventar ciertas restricciones de movimiento, como puede ser el tener que situar los controladores de juego a una determinada altura. Podemos ver un ejemplo de uso en la Figura 2.5.

Figura 2.5: *WalkinVR* - Movimiento y rotación virtual

- **Seguimiento de manos**

  Función destinada a personas que por su patología no puedan sostener los controladores de juego. En la Figura 2.6, podemos observar como a través de un dispositivo *Kinect* se detectan las manos, evitando así el uso de los mandos.

Figura 2.6: *WalkinVR* - Movimiento y rotación virtual

2.4 Comparativa

Una vez comentadas las ideas generales de cada herramienta, se van a destacar aquellas características principales que las diferencian y aquellos puntos que se podrían mejorar en el

desarrollo propuesto en este proyecto.

En el primer estudio (Sección 2.1), el uso de *Leap Motion* en lugar de mandos, proporcionaría un mayor realismo y un amplio abanico de posibilidades en cuanto a la interacción en los escenarios. La diferencia sería notable, pensemos en los movimientos que podemos hacer con las articulaciones de una mano, frente a las limitaciones que tiene el interactuar con un mando como medio.

Por otra, el programar nuestro propio escenario, nos permitiría añadir una opción a la interfaz, en la que poder configurar la inclinación de la cámara encargada de captar los elementos de la escena. De este modo, podríamos adaptar dicha inclinación a la problemática del usuario. En el estudio en cuestión, se evitaría el tener que reclinar la silla de ruedas del usuario.

En el análisis de la herramienta propuesta por la Universidad Politécnica de Barcelona (Sección 2.2), podemos destacar como algo muy positivo el que los escenarios fueran desarrollados por los propios investigadores, esto permite una mejor adaptación de los entornos a las patologías a tratar. Como mejora, en lo que respecta al escenario para tratar problemas de índole cognitivo, se podrían substituir los mandos por un dispositivo *Leap Motion*. No obstante, habría que valorarlo en función del usuario que va a usar dicho escenario, quizás su problemática hace indispensable el uso de controladores para obtener una buena experiencia de juego.

La aplicación *WalkinVR* (Sección 2.3), nos proporciona un amplio abanico de posibilidades, al hacer accesibles todos los juegos del catálogo de *SteamVR* a personas con diversidad funcional. No obstante, como contrapartida, son entornos muy genéricos y que podrían no adaptarse completamente al tratamiento de problemas de rehabilitación físicos o cognitivos.

Capítulo 3

Fundamentos tecnológicos

El presente capítulo tiene como objetivo ofrecer una idea general de cada una de las herramientas o tecnologías analizadas y/o usadas en el desarrollo de este proyecto. Para una mayor claridad, clasificaremos las mismas según pertenezcan a hardware o a software.

3.1 Hardware

3.1.1 HTC VIVE

Es un equipo de realidad virtual creado por *HTC* y *Valve* [7]. Será el encargado de crear la experiencia de realidad virtual, permitiendo la inmersión del usuario en los diferentes escenarios desarrollados.

Los componentes principales de dicho equipo son:

- **Gafas**: Están formadas por una pantalla y una serie de sensores de posición (Figura 3.1). Permite la visualización de entornos virtuales y el seguimiento de la posición del usuario.

  Especificaciones:

  - **Pantalla:** Doble AMOLED 3.6”
  - **Resolución:** 1080 x 1200 píxeles/ojo
  - **Frecuencia de refresco:** 90Hz
  - **Campo de visión:** 110º
  - **Grados de libertad:** 6
  - **Seguridad:** Límites del área de juego y cámara frontal
  - **Sensores:** Seguimiento SteamVR, sensor G, giroscopio y proximidad
  - **Conexiones:** HDMI, USB 2.0, entrada de audio 3.5mm y Bluetooth

Figura 3.1: *HTC VIVE* - Gafas

- **Mandos**: Permiten al usuario interactuar con los diferentes elementos de un escenario, bien mediante el movimiento espacial de los mismos o a través de los botones que incorpora (Figura 3.2). Al igual que las gafas, cuenta con varios sensores que rastrean la posición de las manos.

  Especificaciones:

  - **Sensores:** Seguimiento de SteamVR
  - **Entradas:** Trackpad multifunción, botones de agarre, disparador de dos etapas, botón de sistema y botón de menú
  - **Duración de la batería:** 6 horas aproximadamente
  - **Conexiones:** Puerto de carga micro-USB

Figura 3.2: *HTC VIVE* - Mandos

- **Sensores de posición**: Son los encargados de generar el área de juego y de realizar el seguimiento de gafas y mandos (Figura 3.3).

Figura 3.3: *HTC VIVE* - Sensores de posición

Adicionalmente, se necesita un equipo informático que cumpla con los requisitos mínimos especificados por el fabricante y preparar el espacio físico por donde nos moveremos. Se requiere un área de un mínimo de 3m² y de un máximo de 15m², en donde se colocarán los sensores de posición.

Los requisitos mínimos de dicho equipo son:

- **Procesador**: Intel Core i5-4590 / AMD FX 8350
- **GPU**: NVIDIA GeForce GTX 970 / AMD Radeon R9 290
- **Memoria RAM**: 4 GB
- **Salida de video**: HDMI 1.4 / DisplayPort 1.2
- **Puertos USB**: 1 USB 2.0
- **Sistema Operativo**:Windows 7/ Windows 8 / Windows 10

Una vez hemos instalado todos los componentes, simplemente necesitamos ejecutar un juego en el ordenador, colocar las gafas, agarrar los mandos y situarnos en el área de juego (Figura 3.4). En este proyecto se va a desarrollar un juego o escenario propio, pero también es posible obtener juegos tanto de pago como gratuitos, de plataformas como *Steam VR* o *Viveport*.

Figura 3.4: *HTC VIVE* - Ejecución de un juego

3.1.2 Adaptador inalámbrico VIVE

Originalmente para usar un equipo *HTC VIVE*, es necesario conectar las gafas a nuestro ordenador mediante tres cables (HDMI, USB y alimentación) utilizando un adaptador (Figura 3.5).

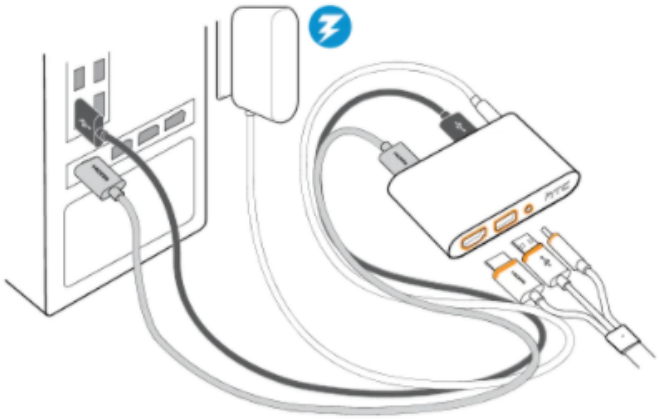

Figura 3.5: *HTC VIVE* - Conexión predeterminada

Esto puede afectar negativamente a la experiencia de uso del entorno de realidad virtual, ya que puede limitar ciertos movimientos o provocar que tropecemos con dichos cables.

Para el desarrollo de este trabajo, se cuenta con un adaptador inalámbrico (Figura 3.6), creado por el propio fabricante de *HTC VIVE* para solventar estos inconvenientes.

Figura 3.6: *HTC VIVE* - Equipo de conexión inalámbrica

3.1.3 Correa de audio VIVE

El sonido es otro elemento fundamental para crear un entorno de realidad virtual de máximo realismo. Las gafas *HTC VIVE* cuentan con una salida de audio en donde podemos conectar unos auriculares.

También es posible adquirir un complemento de audio que se integra totalmente en la estructura de las gafas. Este dispositivo cuenta con dos altavoces regulables en altura y ángulo (Figura 3.7).

El uso de este complemento evita el tener que usar unos auriculares como accesorio externo y la posibilidad de desconexión o caída de los mismos durante la ejecución de un juego.

Contamos con este dispositivo para el desarrollo de este proyecto.

Figura 3.7: *HTC VIVE* - Correa de audio

3.1.4 Leap Motion

Leap Motion [4] es un sistema óptico capaz de realizar un seguimiento de nuestras manos y dedos (Figura 3.8), permitiendo la interacción con contenido digital como, por ejemplo, juegos o aplicaciones de diferente índole.

Especificaciones:

- **Conexiones:** USB 2.0
- **Método de montaje:** Escritorio o visor de realidad virtual
- **Zona de interacción:** 140 x 120 grados
- **Cámaras:** 2 cámaras infrarrojas de 640x240 píxeles
- **Requisitos mínimos para su uso en Escritorio:** Windows 7+, AMD Phenom II/Intel Core I3, 2 GB RAM

Este dispositivo se integra perfectamente en las gafas *HTC VIVE* (Figura 3.9). Pensando en los objetivos del presente proyecto, el uso de *Leap Motion* aporta grandes beneficios en la interacción con los elementos de los diferentes escenarios. Además de proporcionar entornos mucho más realistas, no es comparable la cantidad de movimientos que podemos realizar con las manos a las que nos permite realizar un mando. Por todo ello, se decide hacer uso de dicho dispositivo en todos los escenarios a desarrollar.

Figura 3.8: *Leap Motion* - Aplicación de escritorio

Figura 3.9: *HTC VIVE* con *Leap Motion*

3.2 Software

3.2.1 Unity

Unity [8] es una plataforma de desarrollo 3D en tiempo real, que permite crear entornos interactivos para múltiples plataformas como PC, consolas o equipos de realidad virtual [9].

Está disponible con sistemas operativos *Windows*, *Mac OS* o *GNU/Linux* y ofrece diversos tipos de licencias clasificadas según se vaya a usar en un entorno personal o empresarial.

Las licencias de uso personal son totalmente gratuitas siempre y cuando no se superen unos ingresos de 100.000$ anuales. Dentro de este tipo de licencia, encontramos las licencias personales o de estudiante, muy similares entre sí. No obstante, para el desarrollo de este proyecto hacemos uso de una licencia de estudiante ya que proporciona acceso a *Unity Learn*

Premium, un catálogo con múltiples tutoriales para familiarizarse con el editor de *Unity*.

Fundamentalmente, la metodología de trabajo en *Unity* se descompone en dos partes:

- **Modelado 3D:** Hace referencia a la creación o importación de objetos 3D. En *Unity* serán representados mediante elementos denominados *GameObjects*, que por medio de componentes o scripts definirán la apariencia y comportamiento de esos objetos 3D.
- **Programación:** Es la parte que dota de funcionalidad a los *GameObjects*, en forma de componentes o scripts. Un script no es más que un componente creado por nosotros en un lenguaje de programación. En concreto, para programar scripts en *Unity* tendremos que usar *C#*.

3.2.2 C#

C# [10] es un lenguaje de programación desarrollado por *Microsoft*. Deriva de la familia de lenguajes *C*, por lo que su sintaxis es muy similar. Es un lenguaje orientado a componentes. Los lenguajes orientados a componentes presentan conceptos, metodologías o estándares muy similares a los lenguajes orientados a objetos, no obstante los primeros presentan un mayor nivel de abstracción [11] .

3.2.3 Microsoft Visual Studio

Microsoft Visual Studio [12] es un entorno de desarrollo compatible con múltiples lenguajes de programación como *C#*, *C++*, *Python* o *JavaScript*.

Se elige dicho IDE por poseer una alta integración con *Unity*, facilitando en gran medida la parte de programación de nuestros escenarios.

Para tener acceso a dicho programa de forma gratuita se hace uso de la licencia de estudiante proporcionada por la Universidad.

3.2.4 OpenVR

OpenVR [13] es un API creado por *Valve* que permite el acceso al hardware de diferentes equipos de realidad virtual como, por ejemplo, *HTC VIVE* o *Oculus Rift*.

Es necesario añadir *OpenVR* en *Unity* para compilar correctamente nuestro entorno y, obtener así, un programa funcional para el equipo *HTC VIVE*.

3.2.5 SteamVR y Steam

SteamVR [14] es el entorno de ejecución que permitirá usar la aplicaciones desarrolladas con el API *OpenVR*. Dicha aplicación se distribuye con *Steam*, una plataforma de distribución digital de videojuegos desarrollada por *Valve* [15]. Por lo tanto, será necesario instalar

primeramente *Steam* en el ordenador. Una vez hecho esto, podemos instalar *SteamVR* manualmente; de lo contrario, se instalará automáticamente cuando *Steam* detecte un equipo de realidad virtual.

3.2.6 Plugin SteamVR

Unity permite la instalación de múltiples complementos o plugins, uno de los más interesantes en lo que concierne a este proyecto es *SteamVR Unity Plugin* [16]. Este complemento hace uso del API *OpenVR* y proporciona numerosas herramientas y componentes prefabricados que facilitan el desarrollo de escenarios de realidad virtual. Su handicap es que los escenarios resultantes solo podrán ser ejecutados en las plataformas compatibles con *OpenVR*.

3.2.7 XR Interaction Toolkit

XR Interaction Toolkit [17] es un paquete creado por *Unity* que proporciona un conjunto de herramientas para crear entornos de realidad virtual con relativa facilidad. A diferencia de *SteamVR Unity Plugin*, los entornos creados son totalmente independientes de la plataforma de destino.

3.2.8 Controlador Leap Motion

Para poder utilizar el sistema *Leap Motion*, es necesario instalar su controlador [18] en el equipo en donde se ejecute el entorno de realidad virtual.

3.2.9 Modulos Unity Leap Motion

El fabricante de *Leap Motion* facilita una serie de módulos [19] que deben ser importados en *Unity* con la finalidad de tener acceso al API del dispositivo, así como a diferentes recursos que facilitaran el desarrollo de los diferentes escenarios. En concreto, en este TFG, se hacen uso de los módulos *Core* y *Interaction Engine*.

- **Core:** Incluye los componentes necesarios para permitir la interacción entre los objetos del escenario y el dispositivo *Leap Motion*.
- **Interaction Engine:** Contiene diferentes representaciones físicas de manos y proporciona los controladores para permitir diferentes acciones con las mismas como puede ser el agarre o contacto con objetos.

Capítulo 4

Metodología

PARA el desarrollo del presente proyecto se opta por una metodología ágil, adaptando aspectos de una de las metodologías más utilizadas, *Scrum*.

En este capítulo se expondrán las características y ventajas del uso de metodologías ágiles, se darán unas nociones básicas de *Scrum* y se expondrá la adaptación de la metodología a este proyecto.

4.1 Metodología ágil

Una metodología ágil es aquella que permite adaptar el modo de trabajo a las condiciones del proyecto, aportando flexibilidad y eficiencia y, por tanto, obteniendo un mejor producto a menor coste (Figura 4.1). Esta metodología surge tras la búsqueda de alternativas al método tradicional de trabajo, muy estructurado y estricto, basado en un modelo en cascada [20] .

Figura 4.1: Metodología Ágil

En Febrero de 2001, se reunieron, entre otras personalidades, reconocidos desarrolladores de software con la finalidad de discutir métodos de desarrollo de software ágiles. De esta reunión y de otros debates surgieron una serie de principios que dieron lugar al *Manifiesto Ágil*, que estableció las bases que debe cumplir cualquier método ágil [21].

Los principios que reza dicho manifiesto son:

- Individuos e interacciones sobre procesos y herramientas
- Software funcionando sobre documentación extensiva
- Colaboración con el cliente sobre negociación contractual
- Responder ante el cambio sobre seguir un plan

4.2 Scrum

Scrum es una de las metodologías ágiles más utilizadas en el mundo. Se basa en realizar entregar parciales y regulares del producto final, priorizadas por el beneficio que aportan al cliente. Es una metodología altamente recomendada en proyectos con entornos complejos, en los que se necesitan resultados pronto y donde los requisitos son cambiantes o poco definidos [22].

Dentro del modelo de trabajo de *Scrum* (Figura 4.2) distinguimos roles, hitos y herramientas [23].

4.2.1 Roles

El equipo humano participante en el desarrollo del proyecto tendrá uno de los siguientes roles:

- **Product Owner:** Se encarga de maximizar el valor del trabajo del equipo de desarrollo. El maximizar en mayor o menor medida dicho valor, será consecuencia de la gestión del conjunto de tareas a realizar en el proyecto, conocido dentro de la terminología de *Scrum* como *Product Backlog*.
- **Scrum Master:** Es el responsable de que se apliquen los principios de la metodología en la organización y de solventar aquellos problemas que puedan surgir en cada iteración o Sprint.
- **Equipo de desarrollo:** Es el conjunto de profesionales encargados del desarrollo del producto. Es un equipo multifuncional y autoorganizado que estará compuesto por entre 5-7 personas.

4.2.2 Eventos

El desarrollo del producto se organiza en iteraciones o sprints que contienen una serie de eventos. Estos eventos son:

- **Sprint:** Es el núcleo de *Scrum* y contendrá el resto de eventos del proceso. Su duración estimada es de entre 2 y 4 semanas. En cada Sprint, el equipo de desarrollo debe construir un *incremento del producto*, una entrega parcial que consiste en una versión del producto realizada durante el proceso de desarrollo. Cada incremento debe ser funcional y mostrar una mejoría respecto al anterior.

- **Sprint planning:** Es una reunión en la que se definen las tareas y objetivos a cumplir en el Sprint. El *Product Owner* presenta los requisitos del producto, ordenados por su prioridad. Se resuelven posibles dudas y el equipo de desarrollo selecciona los requisitos que se incluirán en el presente Sprint. Además, los miembros de dicho equipo crearán una lista con las tareas a realizar en el Sprint y se organizarán para el desarrollo eficiente de las mismas.

- **Daily meeting:** Es una reunión diaria de un máximo de 15 minutos de duración. En ella, el equipo pone en común el trabajo realizado, el progreso del Sprint y posibles dificultades que estén ralentizando la realización de las tareas. Es indispensable que participen el equipo de desarrollo y el *Scrum Master*.

- **Sprint review:** Es una reunión que se realiza al final de cada Sprint. En ella, el *Product Owner* presenta el incremento del producto al cliente y el equipo de desarrollo muestra su funcionamiento. El cliente valida los cambios realizados y es posible que solicite modificaciones o mejoras que el *Product Owner* deberá añadir al *Product backlog*.

- **Sprint retrospective:** Es el último evento del Sprint. Es una reunión en la que se evalúa cómo se ha implementado la metodología durante el desarrollo de esa iteración y se proponen posibles mejoras a implementar en el siguiente Sprint.

4.2.3 Herramientas

Las herramientas propuestas en *Scrum* están orientadas a ofrecer la información con la máxima transparencia dentro del equipo de trabajo. Aquí tendremos:

- **Product backlog:** Representa el conjunto de tareas que es necesario realizar para completar el proyecto en su totalidad. Será responsabilidad del *Product Owner* el establecer prioridades a la hora de completar estas tareas.

- **Sprint backlog:** Contiene aquellas tareas del *Product backlog* que se han seleccionado durante el *sprint planning* para llevar a cabo en el Sprint.

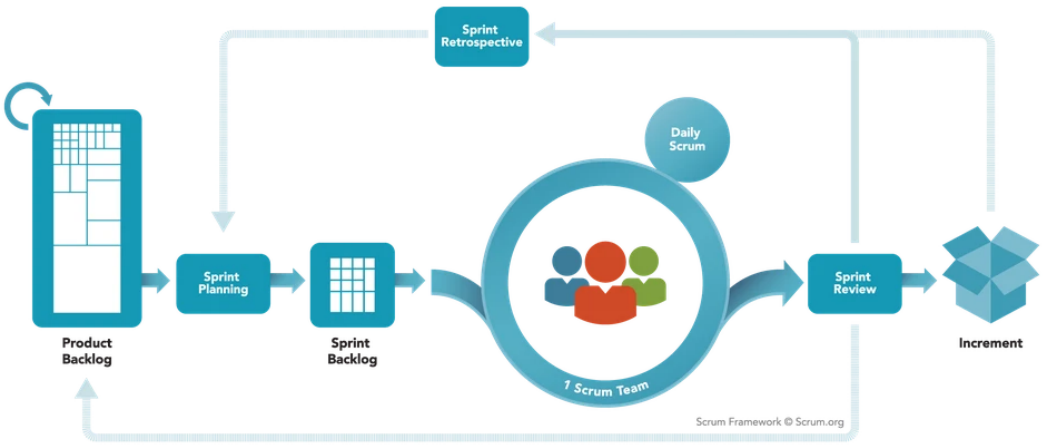

Figura 4.2: Marco de trabajo en Scrum

4.3 Adaptación de la metodología al proyecto

Como se indicaba al comienzo de este capítulo, para el desarrollo de este trabajo se ha seguido una metodología ágil, en concreto *Scrum*, pero adaptando ciertos aspectos a las características del proyecto.

En cuanto a la repartición de roles, la metodología está pensada para el trabajo en equipos de 5-7 personas, algo que no procede en el desarrollo de un TFG. De modo que, el rol de *Product Owner* fue asumido por los directores del proyecto, mientras que los roles de *Scrum Master* y de equipo de desarrollo fueron llevados a cabo por el alumno.

La duración de los Sprints varió entre 2 y 4 semanas, debido a periodos en los que el alumno no podía dedicarse por completo al desarrollo del proyecto.

En la primera reunión, los directores expusieron el *Product backlog*, indicando las prioridades de cada una de las tareas. Este sufrió diversas modificaciones a lo largo del proyecto, en las que se añadieron nuevas funcionalidades o mejoras.

Los eventos se unificaron en una sola reunión por Sprint, en la que participaban directores y alumno. Al comienzo de cada reunión, se revisaba el Sprint anterior, comprobando el correcto funcionamiento del incremento del producto y proponiendo correcciones o mejoras. Posteriormente, se definían las tareas y objetivos a cumplir en el siguiente Sprint (*Sprint backlog*).

Algunas reuniones no eran necesarias o se podían reducir ya que el equipo era de una única persona. Además, parte de las reuniones no se pudieron realizar de forma presencial o en las fechas previstas. No obstante, se ha mantenido la comunicación de forma telemática, empleando *Teams* o correo electrónico.

Capítulo 5

Planificación y estimación de costes

En este capítulo se expondrá la planificación, los recursos empleados y la estimación del coste total del proyecto.

5.1 Planificación

Como resultado de haber aplicado la metodología explicada en el Capítulo 4, en el desarrollo completo del proyecto se han definido las siguientes iteraciones:

- **Sprint 0:** Explicación de los objetivos del proyecto, primera toma de contacto con el equipo de realidad virtual y análisis de las diferentes tecnologías.
- **Sprint 1:** Formación en el uso de *Unity*.
- **Sprint 2:** Realización de un escenario de prueba que permita la interacción con dos cubos a través de un ratón y de los mandos *HTC VIVE*.
- **Sprint 3:** Modificación del escenario realizado en el Sprint 2, permitiendo la interacción con las manos haciendo uso del dispositivo *Leap Motion*.
- **Sprint 4:** Creación del escenario *Grifo*.
- **Sprint 5:** Creación de los escenarios *Cubiertos*, *Frutero* y *Menu*, interfaz de usuario y exportación de datos de juego.
- **Sprint 6:** Creación del escenario *Desayuno*, incorporación de señales visuales y de audio para lograr mayor realismo y proporcionar *feedback* al usuario e integración de los escenarios en entornos reales.
- **Sprint 7:** Realización de la memoria y entrega del producto para su aprobación.

Como se ha indicado anteriormente, la duración de cada Sprint no fue uniforme ya que el alumno no pudo dedicarse a tiempo completo en todas las etapas del desarrollo. Así mismo, se ha realizado una parada en el proyecto entre los Sprints 0 y 1, por priorizar la entrega de trabajos y la preparación de los estudios de grado.

También se quiere destacar que los conocimientos que se fueron adquiriendo en cada Sprint provocaron una mayor eficiencia en la realización de los Sprint finales, acortando significativamente los tiempos de duración.

En la Tabla 5.1 se pueden visualizar las fechas de inicio y fin de cada Sprint.

Sprint	Inicio	Fin
0	24/03/2020	24/04/2020
1	05/06/2020	09/07/2020
2	09/07/2020	30/07/2020
3	30/07/2020	21/08/2020
4	21/08/2020	25/09/2020
5	25/09/2020	22/10/2020
6	22/10/2020	05/11/2020
7	05/11/2020	16/11/2020

Tabla 5.1: Fechas de inicio y fin de cada Sprint

A través del Diagrama de Gantt (Figura 5.1), podemos visualizar la parada realizada en el proyecto entre el Sprint 0 y el Sprint 1. Por otra parte, se puede distinguir que la duración de los Sprints 0, 1 y 4 destaca sobre el resto, el motivo es que dichas iteraciones implicaron tareas de análisis y formación, que provocaron una alta carga de trabajo y, en consecuencia, una mayor dedicación.

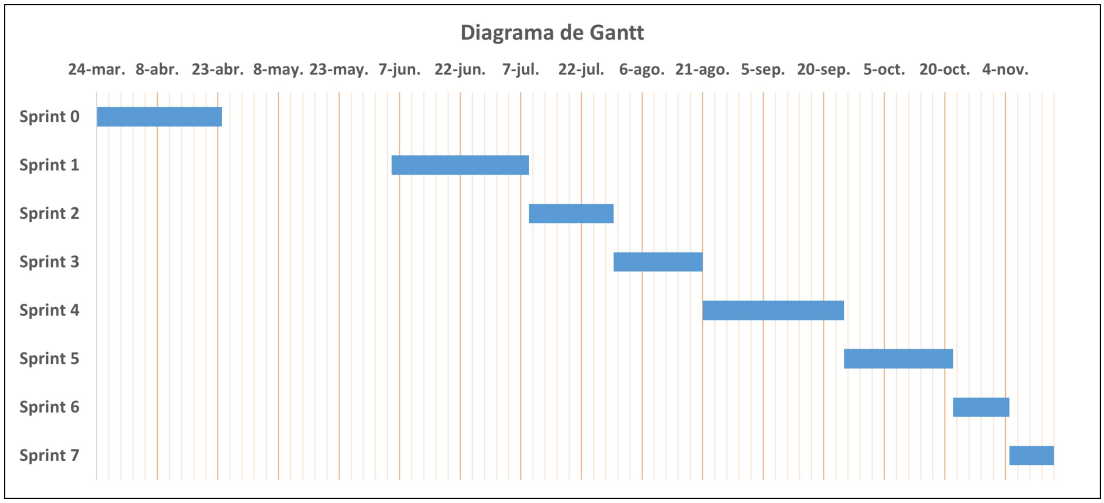

Figura 5.1: Diagrama de Gantt

5.2 Recursos

5.2.1 Hardware

- Ordenador personal del alumno
- Ordenador para la ejecución de los entornos
- Equipo *HTC VIVE* y complementos
- Dispositivo *Leap Motion*

5.2.2 Software

- Unity 2019.4.5.5f1
- Microsoft Visual Studio 2019

5.2.3 Humanos

En un proyecto real, el equipo humano estaría formado por:

- Jefe de proyecto
- Analistas
- Diseñadores Software
- Diseñadores gráficos
- Desarrolladores
- Testers

Dadas las características del proyecto, el papel de jefe de proyecto fue asumido por los directores del proyecto y el resto de roles recayeron en el alumno.

5.3 Coste

Todo el hardware necesario fue proporcionado por el CITIC[1], de modo que no supuso ningún coste para el proyecto.

En cuanto al software, tanto en *Unity* como en *Microsoft Visual Studio*, se han utilizado licencias de estudiante, sin coste.

[1] Centro de Investigación en Tecnologías de la Información y las Comunicaciones

Por lo tanto, el coste total del proyecto sería el correspondiente al equipo humano. Se estimó un coste de 30€/hora para el alumno y un coste de 50€/hora para cada director [24].

En la Tabla 5.2, se desglosa el número de horas dedicadas por el alumno y los directores a cada Sprint.

Sprint	Alumno	Director 1	Director 2
0	124	4	4
1	136	4	4
2	84	4	4
3	88	4	4
4	140	4	4
5	108	4	4
6	56	4	4
7	44	4	4
Total	780	32	32

Tabla 5.2: Horas dedicadas por Sprint

En la Tabla 5.3, podemos visualizar el coste total del proyecto tras sumar los costes del alumno y de los directores.

Recurso	Horas	Coste (€/hora)	Coste (€)
Alumno	780	30	23.400
Director 1	32	50	1600
Director 2	32	50	1600
Coste total del proyecto			26.600 €

Tabla 5.3: Coste del proyecto

Capítulo 6

Desarrollo

En este capítulo se describirán las etapas que formaron parte del desarrollo del proyecto, esto es, análisis, diseño, implementación y pruebas.

6.1 Análisis

El análisis es una de las etapas más importantes en el desarrollo de un proyecto software. Su finalidad es comprender completamente los requisitos de la herramienta a desarrollar, evitando ambigüedades o funcionalidades mal definidas.

En lo que respecta a este proyecto, se ha realizado una etapa de análisis previa, en cada una de las iteraciones que implicaban desarrollo software. Así, antes de desarrollar cada entorno de realidad virtual, se estudiaron los requisitos y funcionalidades que debía cumplir, obteniendo así un conjunto de requerimientos totalmente definidos.

6.2 Diseño

Tras la etapa de análisis, en el desarrollo habitual de un proyecto software se valoraría el uso de un patrón de diseño y se crearían los diagramas de clases.

Teniendo en cuenta la metodología de trabajo en *Unity*, explicada en la Sección 3.2 del Capítulo 3, no procede crear una jerarquía de clases o aplicar un patrón de diseño. Este proceso será sustituido por la elección de los componentes más apropiados para dotar de funcionalidad a los diferentes *GameObjects* de la escena. Como se ha comentado, las funcionalidades de dichos componentes son limitadas y, en ocasiones, será necesario desarrollar nuestros propios componentes mediante scripts. De forma previa a la implementación de un script, se creará un boceto para definir las variables, funciones y métodos que deberá contener. Así mismo, se ha realizado un diagrama de flujo que representa la transición entre las diferentes escenas de la aplicación (Figura 6.1).

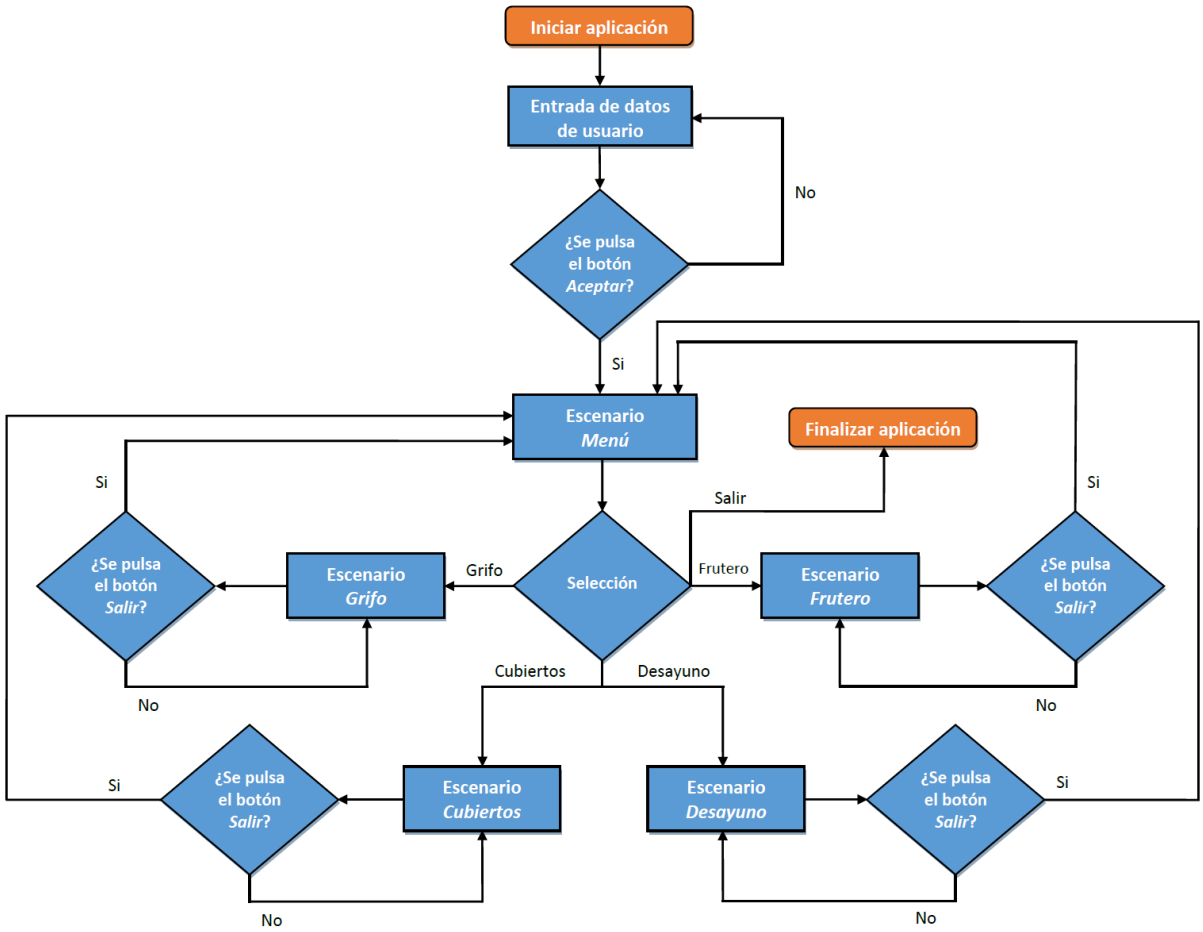

Figura 6.1: Diagrama de flujo de transición entre escenas

6.3 Implementación

6.3.1 Sprint 0

Fecha: 24/03/2020-24/04/2020

En esta primera iteración, se realizó la primera reunión del proyecto. En ella, los directores expusieron los requisitos que se deberían cumplir y se explicó la metodología de trabajo. Además, se hizo una primera toma de contacto con el equipo de realidad virtual probando diversos juegos y aplicaciones.

Por otra parte, se empezaron a analizar las diferentes tecnologías que se utilizarían en el desarrollo del producto.

Por la incertidumbre actual, a causa del virus *COVID-19*, se valoró el facilitar el equipo de realidad virtual al alumno para que pudiera ejecutar las pruebas en su domicilio, sin necesidad de acudir al CITIC.

Para ello, se ejecutó un software de *HTC* [25] que comprueba las características hardware del equipo informático e indica su compatibilidad con las gafas *HTC VIVE*.

El resultado de estas pruebas de compatibilidad indicaba que era posible que el equipo de realidad virtual no funcionase correctamente en nuestro equipo.

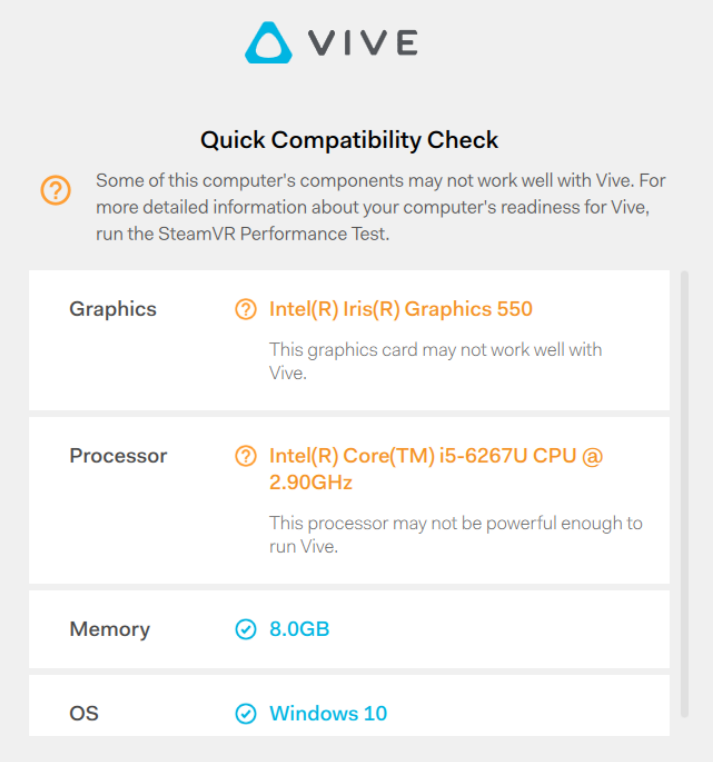

Figura 6.2: Test compatibilidad *HTC VIVE*

Tras comprobar que el equipo del alumno no era compatible, se valoró el uso de un equipo de realidad virtual autónomo, es decir, aquel que no depende de un ordenador para ejecutar los juegos o escenarios. También se buscaron otros equipos que, aun no siendo autónomos, no precisaran de unas características hardware tan elevadas.

En la Tabla 6.1, se pueden observar de forma resumida los resultados de analizar los principales dispositivos del mercado [7] [26] [27].

Modelo	¿Autónomo?	Características principales	Requisitios mínimos	Coste
HTC VIVE	No	- Pantalla Doble AMOLED - 1080 x 1200 píxeles/ojo - 90Hz - 110º de campo de visión - 6º de libertad - 2 mandos	- NVIDIA GeForce GTX 970 - 4 GB de RAM - Intel Core i5-4590 - Windows 7	638,99€ [28]
Oculus Rift S	No	- Pantalla LCD - 1280 x 1440 píxeles/ojo - 80Hz - 6º de libertad - 2 mandos	- NVIDIA GTX 1050 Ti - 8 GB de RAM - Intel Core i3-6100 - Windows 10	449€ [29]
Valve Index	No	- Pantalla Doble LCD - 1440 x 1600 píxeles/ojo - 120Hz -130º de campo de visión - 6º de libertad - 2 mandos	- NVIDIA GeForce GTX 970 - 8 GB de RAM - Dual Core con Hyper-Threading - Windows 10	1079€ [30]
Oculus GO	Si	- Pantalla LCD - 1280 x 1440 píxeles/ojo - 60Hz - 3º de libertad - 1 mando		275€ [31]
Oculus Quest	Si	- Pantalla OLED - 1440 x 1600 píxeles/ojo - 72Hz - 6º de libertad - 2 mandos		469€ [32]

Tabla 6.1: Análisis de equipos de realidad virtual

Tras una valoración de los equipos anteriores, se llegó a la conclusión de que el equipo ya disponible en el CITIC era el más idóneo para lograr los objetivos que se pretendían con el producto final. Por lo tanto, se realizó el máximo trabajo telemáticamente y se acudió al CITIC cuando fue indispensable realizar pruebas con el equipo.

6.3.2 Sprint 1

Fecha: 05/06/2020-09/07/2020

En esta iteración, se profundizó en el aprendizaje y uso de Unity. Para ello, se realizaron diversos tutoriales del programa *Unity Learn* [33] y se leyeron los apartados principales del manual de *Unity* [34] .

Primeramente, se siguieron una serie de guías para familiarizarse con la interfaz. El editor de *Unity* presenta una estructura como la mostrada en la Figura 6.3.

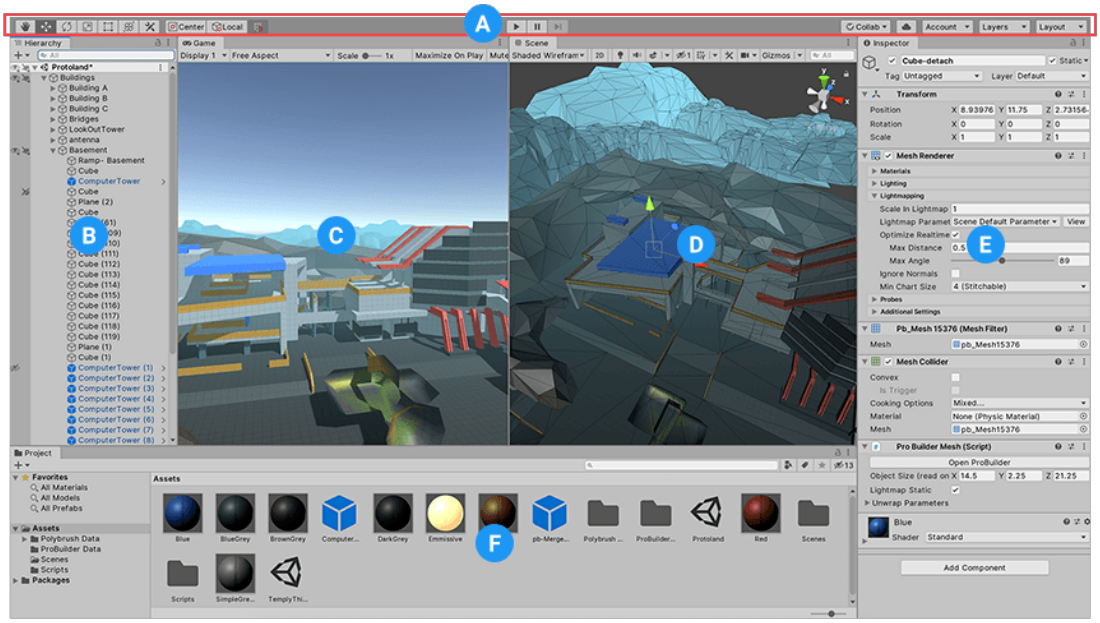

Figura 6.3: Vista del editor de *Unity*

Los elementos principales que se pueden distinguir y con los que se trabajó en el desarrollo de los diferentes entornos son:

- **(A) Barra de herramientas:** Proporciona acceso a las funciones principales que permiten configurar los objetos en la escena. Además, también nos permitirá ejecutar el entorno de juego a través del botón *Play*.
- **(B) Jerarquía:** Representa jerárquicamente todos los objetos de la escena.
- **(C) Vista de juego:** Permite ver la imagen captada por la cámara del juego una vez se pulsa el botón *Play*.
- **(D) Vista de escena:** Representa el escenario sobre el cual colocaremos los diferentes objetos.
- **(E) Inspector:** Permite ver y configurar las propiedades de los componentes de cada objeto, así como añadir nuevos componentes.
- **(F) Proyecto:** Muestra todos los recursos disponibles para usar en el proyecto.

A continuación, se realizaron diversos tutoriales de creación de juegos sencillos (Figura 6.4), pero que ayudaron significativamente a entender el ecosistema de *Unity*.

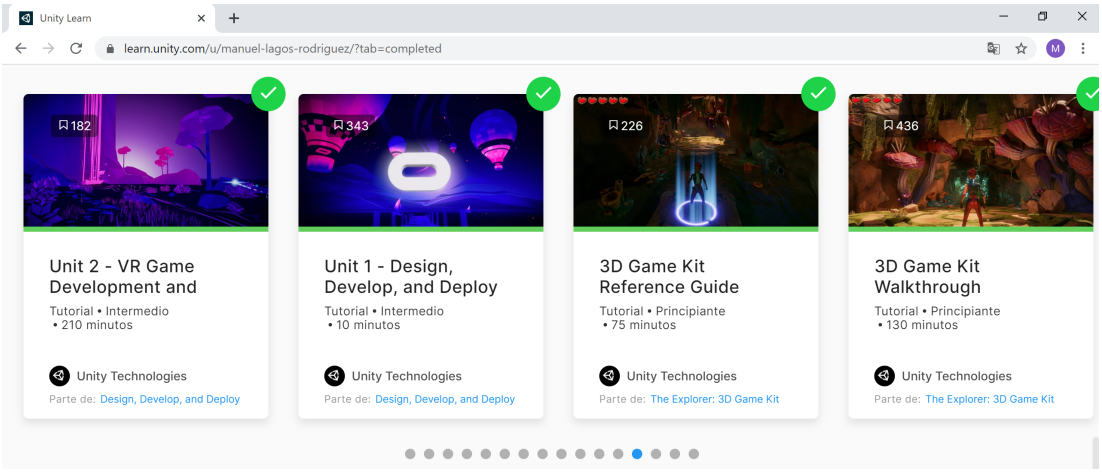

Figura 6.4: Tutoriales *Unity Learn*

En la creación de un escenario de juego en *Unity*, podemos distinguir dos partes principales:

- **Modelado 3D:** Implica la creación o importación de objetos 3D. En *Unity* son representados mediante elementos denominados *GameObjects*. Cada *GameObject* está formado por una serie de componentes que definen la apariencia de dicho objeto.

  Unity únicamente permite la creación de objetos básicos como, por ejemplo, cubos o esferas. No obstante, el diseño de objetos no es su cometido, para ello existen otro tipo de programas específicos.

  Para el desarrollo de los escenarios iniciales se usaron los objetos proporcionados por *Unity*. Sin embargo, para los escenarios finales, con el objetivo de obtener un mayor realismo, se importaron objetos diseñados profesionalmente.

- **Programación:** Una vez tenemos los objetos situados en la escena, es necesario dotarlos de funcionalidad. La programación en *Unity* se realiza en base a scripts, que se añaden a cada objeto en forma de componente. El lenguaje usado para programar dichos scripts es *C#*.

6.3.3 Sprint 2

Fecha: 09/07/2020-30/07/2020

En esta iteración se realizó un primer escenario de prueba que permitía la interacción con dos cubos a través del ratón y de los controladores de *HTC VIVE*.

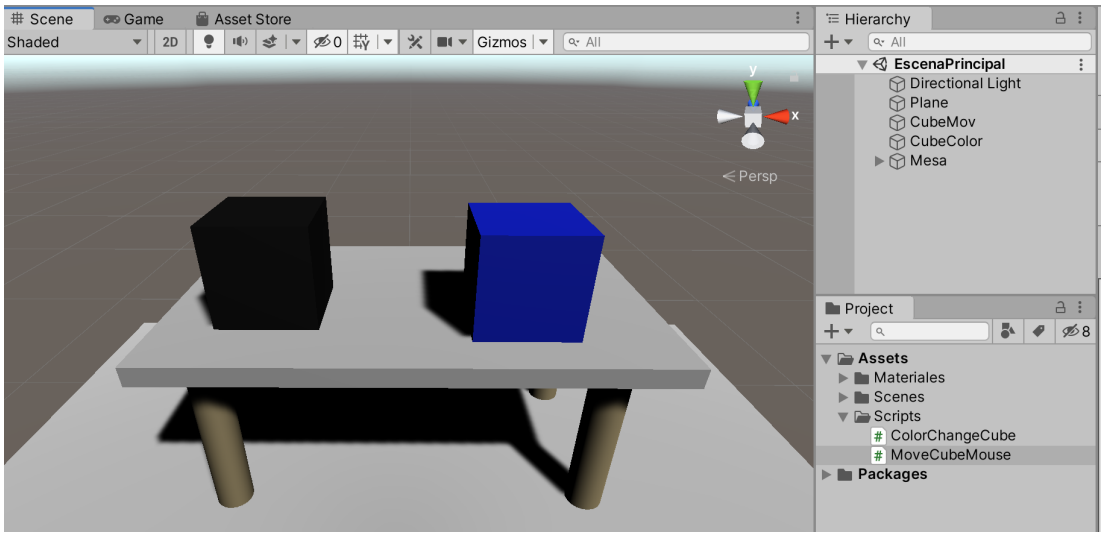

Figura 6.5: Escenario inicial de prueba - Control con ratón

En la Figura 6.5, se pueden observar los elementos de la escena. El cubo negro es el denominado *CubeMov* y el cubo azul se corresponde a *CubeColor*. A continuación, se explica el comportamiento de cada uno.

- **CubeMov:** Se desplaza por el espacio de juego mientras movemos el ratón a la vez que mantenemos pulsado el botón izquierdo del mismo. También es posible moverlo con los mandos del equipo *HTC VIVE*. Para ello, se deberá apuntar al cubo, apretar el gatillo y mover el mando en la dirección deseada.

- **CubeColor:** Su color cambia de azul a rojo al pulsar el botón izquierdo del ratón o bien al apretar el gatillo de un mando del equipo *HTC VIVE*.

Para la interacción con el ratón, se consultó el API de *Unity* y se crearon dos scripts muy sencillos. El script *MoveCubeMouse* se añadió como componente al objeto *CubeMov* y el script *ColorChangeCube* se agregó al objeto *CubeColor*. En las figuras 6.6 y 6.7 se pueden visualizar ambos códigos.

```
using UnityEngine;

public class MoveCubeMouse : MonoBehaviour
{
    private Vector3 mOffset;

    private float mZCoord;

    void OnMouseDown()
    {
        mZCoord =
    Camera.main.WorldToScreenPoint(gameObject.transform.position).z;
        mOffset = gameObject.transform.position -
    GetMouseWorldPos();
    }
    private Vector3 GetMouseWorldPos()
    {
        Vector3 mousePoint = Input.mousePosition;
        mousePoint.z = mZCoord;
        return Camera.main.ScreenToViewportPoint(mousePoint);
    }

    void OnMouseDrag()
    {
    transform.position = GetMouseWorldPos() + mOffset;
    }

}
```

Figura 6.6: Script vinculado a *CubeMov*

```
using UnityEngine;

public class ColorChangeCube : MonoBehaviour
{
    Renderer m_Renderer;

    void Start()
    {
        m_Renderer = GetComponent<Renderer>();
    }

    void OnMouseDown()
    {
        m_Renderer.material.color = Color.red;
    }

    void OnMouseUp()
    {
        m_Renderer.material.color = Color.blue;
    }
}
```

Figura 6.7: Script vinculado a *CubeColor*

Si nos fijamos en los códigos anteriores, ambas clases derivan de la clase *MonoBehaviour*. Esta es una característica de la programación en Unity, toda clase heredará predeterminadamente de *MonoBehaviour*.

Otra característica a destacar es que no se deben utilizar constructores, Unity inicializará automáticamente los diferentes objetos. Los parámetros de inicialización que deseamos configurar deben situarse dentro de la función *Start*, proporcionada por la clase *MonoBehaviour*.

A la hora de programar en Unity, con la finalidad de que nuestros escenarios se ejecuten correctamente es necesario diferenciar una serie de funciones de uso común. A continuación, se da una pincelada de cada una de ellas:

- **Awake():** Se invoca una sola vez, aunque el *GameObject* esté inactivo. Se recomienda para inicializar variables antes de que dicho objeto forme parte del escenario de juego.
- **Start():** Se invoca a continuación de *Awake()*, también se invoca una sola vez, pero es necesario que el objeto esté activo. Es muy útil cuando nos interesa que un determinado componente tenga un valor solo cuando esté activo.
- **Update():** Se invoca una vez por cada fotograma. Se debe usar dicha función para manejar la animación de objetos visuales o para detectar los *inputs*.
- **FixedUpdate():** Similar a *Update()*, pero se invoca con un intervalo de tiempo regular, está indicado para tratar con objetos que implementen física.

Para interactuar con los mandos de *HTC VIVE*, se importó el paquete *XR Interaction Toolkit* que facilita en gran medida la configuración de entornos de realidad virtual.

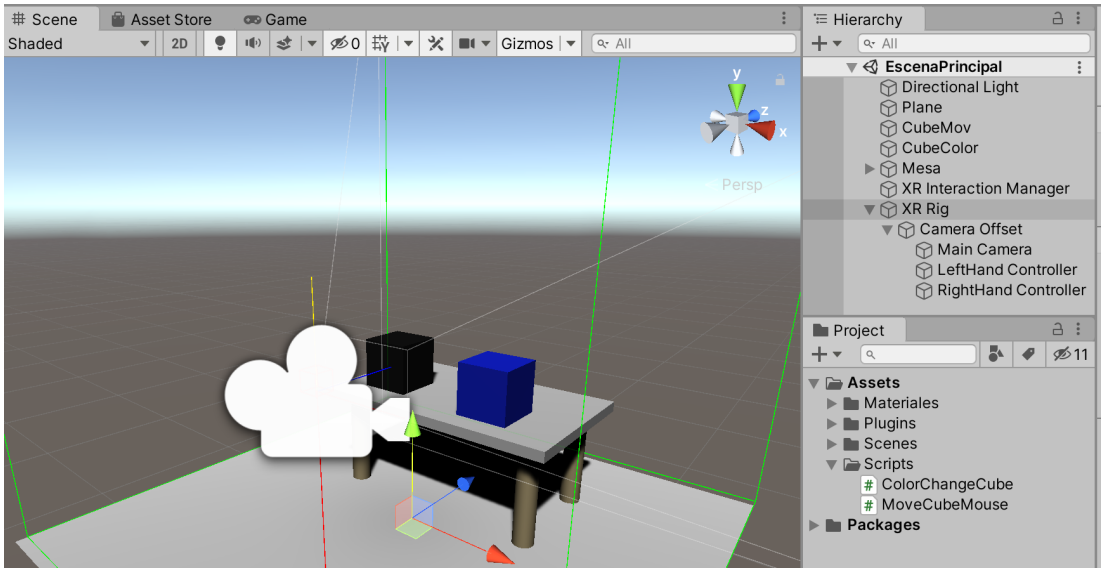

Figura 6.8: Escenario inicial de prueba - Control con mando *HTC VIVE*

Si nos fijamos en la ventana *Hierarchy* en la Figura 6.8, podemos observar los siguientes objetos que permitirán generar un entorno de realidad virtual:

- **XR Interaction Manager:** Es el elemento que actuará de intermediario entre los objetos con los que interactuaremos y los dispositivos con los que actuaremos, es decir, los mandos de *HTC VIVE*.

- **XR Rig:** Es el objeto padre. Contendrá el resto de componentes que configuran el entorno de realidad virtual. Además, permite configurar una serie de parámetros como, por ejemplo, la altura de la cámara conforme a las gafas *HTC VIVE*.

  - **Main Camera:** Es la cámara principal del juego y la única en este caso. Será la encargada de captar los diferentes elementos que se visualizarán en las gafas de realidad virtual.

  - **LeftHand Controller:** Se corresponde al mando izquierdo, en el se configurarán las acciones a realizar al pulsar los diferentes botones.

  - **RightHand Controller:** Tiene la misma funcionalidad que *LeftHand Controller*, pero estaremos configurando el mando derecho.

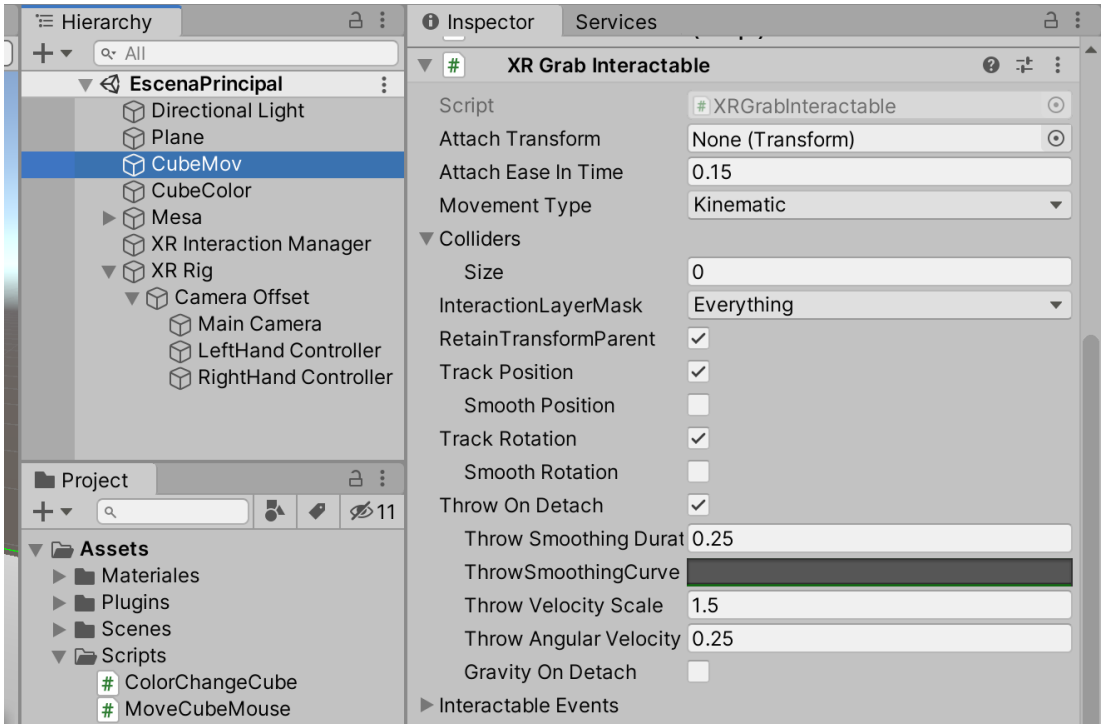

Figura 6.9: Componente *XR Grab Interactable*

Por otra parte, en cada objeto con el que se desee interactuar, tendremos que añadir el componente *XR Grab Interactable* (Figura 6.9). Este componente, cuenta con una serie de parámetros preconfigurados y que para la ejecución de este escenario, han sido totalmente válidos. Además, en dicho componente es donde se indican las acciones a realizar cuando se interactué con alguno de los mandos.

Por último, a la hora de compilar tendremos que añadir el SDK *OpenVR*, obteniendo así un ejecutable para nuestro equipo de realidad virtual.

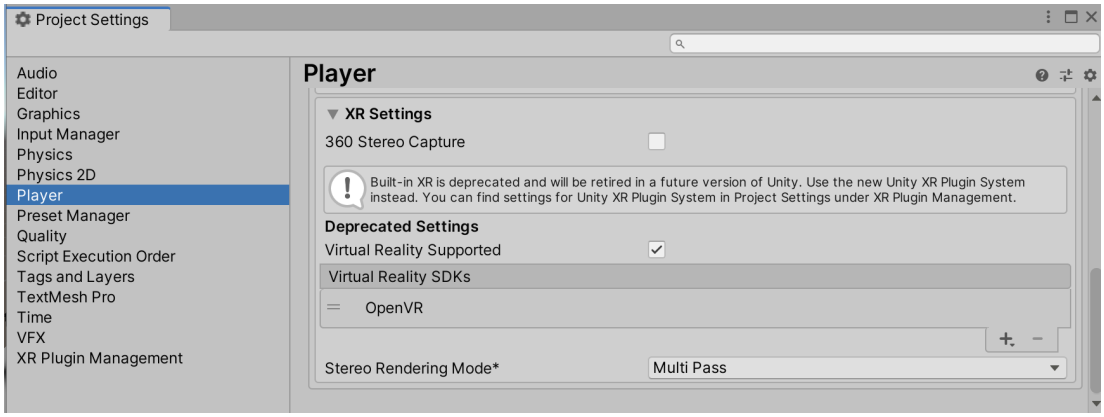

Figura 6.10: Escenario inicial de prueba - Compilación para *HTC VIVE*

Si nos fijamos en la Figura 6.10, se nos indica que la opción elegida está obsoleta. Durante el desarrollo de este proyecto y hasta la actualidad, *Unity* se encuentra en un proceso de cambio en cuanto al modo de compilar los entornos de realidad virtual. Su objetivo es que la compilación se realice mediante la opción *XR Plugin Management* [35] , un sistema basado en el uso de plugins y a través de la cual podemos instalar los plugin de los fabricantes para los que queremos compilar nuestros entornos. El problema es que por el momento no existe un plugin disponible para *OpenVR*, por lo que no nos quedó más remedio que utilizar la opción obsoleta.

6.3.4 Sprint 3

Fecha: 30/07/2020-21/08/2020

En este sprint se implementó la interacción mediante nuestras manos, haciendo uso del dispositivo *Leap Motion*.

Para ello, se hizo uso de una serie de módulos proporcionados por *Leap Motion*. Primeramente, se estudió la documentación [36] para aprender a usar el contenido de dichos módulos. A continuación, se analizaron diferentes escenarios de prueba como el que se puede visualizar en la Figura 6.11 .

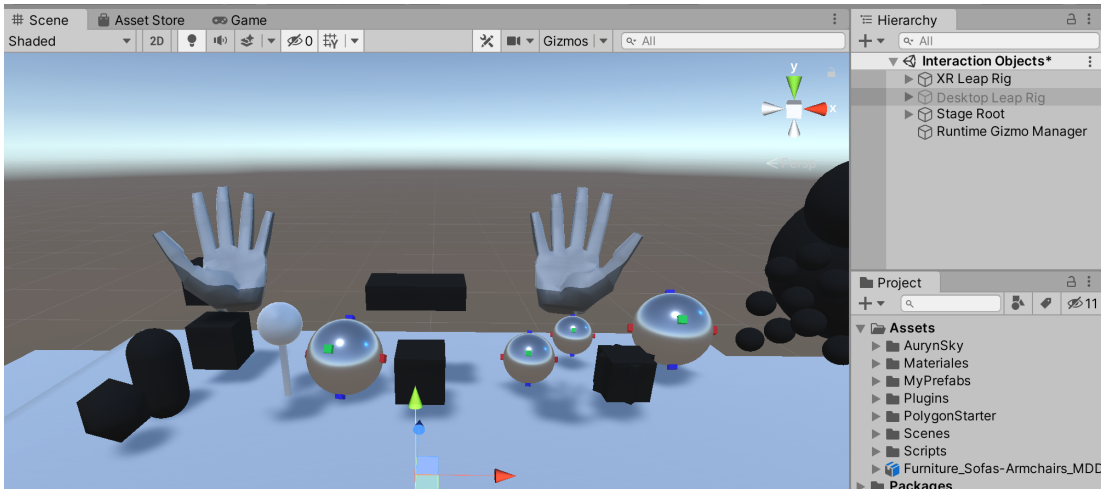

Figura 6.11: Escenario de ejemplo *Leap Motion*

Tras realizar diversos escenarios de prueba y de consultar la documentación junto a los ejemplos facilitados, se consiguió crear un esquema de los *GameObjects* necesarios para integrar el dispositivo *Leap Motion* en un entorno de realidad virtual. Esta tarea supuso un gran peso en el desarrollo del proyecto, pero estableció los pilares para el desarrollo de los escenarios finales.

Los *GameObjects* que usaremos forman parte de los módulos facilitados por *Leap Motion*,

que además de permitirnos la interacción mediante nuestras manos, también incluye los componentes necesarios para crear el entorno de realidad virtual. De modo que, se prescinde del paquete *XR Interaction Toolkit* para el desarrollo de los sucesivos escenarios.

A continuación, se describen los *GameObjects* de este esquema base junto a los componentes que los forman:

- **Leap Rig Base:** Será el objeto padre y contendrá al resto de elementos. Además, cuenta con el componente *XR Height Offset* que permitirá regular diferentes parámetros como la altura de la cámara conforme a las gafas de realidad virtual.
  - **Main Camera:** Se encarga de captar los diferentes objetos de la escena. Tendrá asociado el componente *Leap XR Service Provider*, que se encarga de rastrear nuestras manos.
  - **Hands Models:** Objeto padre que contendrá los modelos 3D de las manos. Además, contará con el componente *Hand Model Manager* que asociará nuestras manos a las manos de la escena.
    - **LoPoly Rigged Hand Left:** Modelo 3D de una mano izquierda completa.
    - **LoPoly Rigged Hand Right:** Modelo 3D de una mano derecha completa.
  - **Interaction Manager:** Es el núcleo de *Leap Motion*, se encarga de la lógica que hace posible la interacción con los objetos de las escenas.
    - **Interaction Hand (Left):** Es el elemento que realmente permite interactuar con nuestra mano izquierda. Cuenta con diferentes parámetros configurables que permitirán, entre otras funciones, seleccionar qué tipos de interacciones se permiten como puede ser agarrar o tocar.
    - **Interaction Hand (Right):** Su funcionalidad es la misma que la de *Interaction Hand (Left)*, pero sobre la mano derecha.
      Además, si se quisiera mantener la interacción con los mandos HTC VIVE, simplemente tendríamos que añadir los elementos *VR Vive-style Controller (Left)* y *VR Vive-style Controller (Right)* que representan a cada uno de los mandos.
- **Interaction Behaviour:** Es el componente que permite la interacción con un objeto. Se debe añadir a cada elemento de la escena con el que queramos interactuar. Contiene una serie de parámetros que podemos modificar para permitir, entre otras opciones, que el objeto solo pueda ser tocado o que sea posible agarrarlo con las dos manos.

En la Figura 6.12, podemos observar el escenario ya configurado, en donde se puede apreciar el componente *Leap Rig Base.* Además, cada cubo contará con el componente *Interaction Behaviour*, para que pueda ser manipulado.

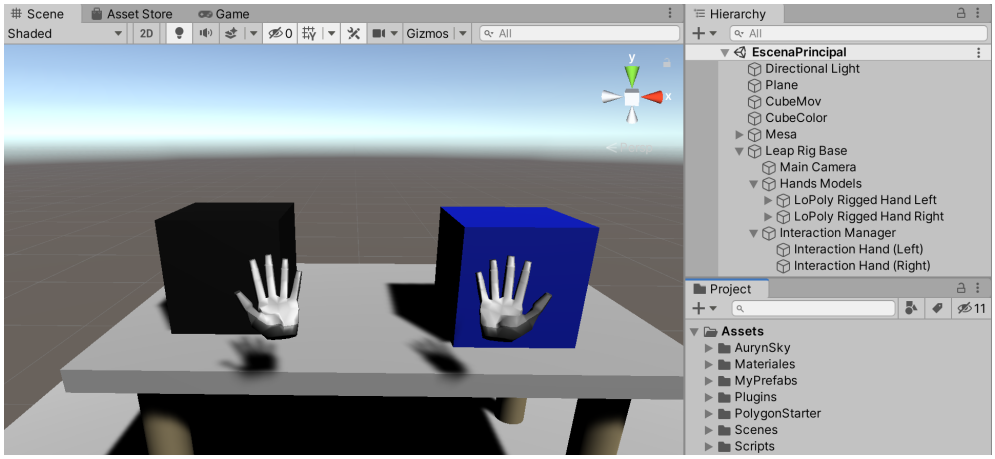

Figura 6.12: Escenario inicial de prueba - Control mediante *Leap Motion*

6.3.5 Sprint 4

Fecha: 21/08/2020-25/09/2020

En esta iteración se desarrolló uno de los entornos fijados en los objetivos del proyecto. Se trata de un grifo cuya apertura implica un movimiento rotacional de la muñeca, uno de los problemas patentes en algunos usuarios. El objetivo de dicha prueba será abrir el grifo, llenar una jarra de agua y volver a cerrar dicho grifo.

Se comenzó por importar un modelo 3D de un grifo y de una jarra, situando ambos objetos en una mesa (Figura 6.13). Posteriormente, con la intención de facilitar el proceso de desarrollo, se configuró el escenario para interactuar mediante un ratón. Por último, se implementó la interacción mediante *Leap Motion*.

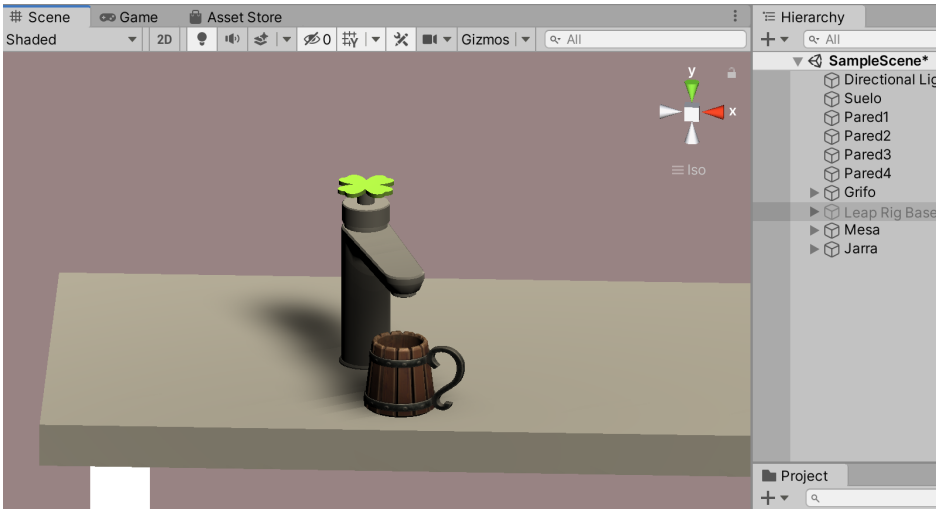

Figura 6.13: Escenario *Grifo*

Para una mayor claridad, se van a explicar de forma separada los dos tipos de control:

- **Control mediante ratón**

  Para dar funcionalidad al escenario, se crea un script mediante el cual se reconocen como entradas la pulsación de los botones izquierdo y derecho del ratón. Se crea un array con una serie de valores ordenados de menor a mayor, que implicarán la velocidad con la que se crean objetos que representan gotas de agua. De modo que, a mayor potencia, se creará un mayor número de objetos y provocará un mayor caudal de agua. Al pulsar el botón izquierdo nos desplazaremos hacia los valores menores de dicho array y si pulsamos el botón derecho, nos desplazaremos en sentido contrario. Además de regular el caudal, cuando se pulsa un botón u otro del ratón se moverá la llave del grifo en un sentido u otro, dependiendo de si estamos abriendo o cerrando.

  Este script se añadirá al objeto *llave* del grifo ya que actuaremos sobre su componente *Transform*, rotando dicho objeto cuando abramos o cerremos el mismo.

  En la Figura 6.14 se puede visualizar la parte del código que detecta la pulsación del ratón y que modifica el valor de un índice que se usará para acceder a los valores del array.

```
1   void OnMouseOver()
2   {
3       if (Input.GetMouseButtonDown(0))
4       {
5           if (indice < potencia.Length-1)
6           {
7               open = true;
8               indice++;
9               GenerateWater();
10          }
11      }
12      else if (Input.GetMouseButtonDown(1))
13      {
14          if (indice > 0)
15          {
16              close = true;
17              indice--;
18              GenerateWater();
19          }
20      }
21  }
```

Figura 6.14: Función input ratón

En la Figura 6.15 podemos observar la función *FixedUpdate()*, de la que se ha hablado anteriormente y dentro de la cual colocaremos aquel código que provocará que la llave gire. Como se puede ver, se hace uso de una variable booleana para identificar el sentido de giro.

```
1   void FixedUpdate()
2   {
3       Vector3 vClose = new Vector3(0, 0, 300);
4       Vector3 vOpen = new Vector3(0, 0, -300);
5
6
7       if (open == true)
8       {
9           rb.AddRelativeTorque(vOpen, ForceMode.Acceleration);
10          open = false;
11      }
12
13      if (close == true)
14      {
15          rb.AddRelativeTorque(vClose, ForceMode.Acceleration);
16          close = false;
17      }
18  }
```

Figura 6.15: Función para rotar el objeto *llave*

En la Figura 6.16, se pueden visualizar las funciones que controlaran y generarán el flujo de agua. Cabe destacar que *drop* es el objeto que representa una gota de agua.

```
1   void GenerateWater()
2   {
3       float showindice = potencia[indice];
4       Debug.Log(showindice);
5       CancelInvoke("FlujoAgua");
6       InvokeRepeating("FlujoAgua", 0, potencia[indice]);
7   }
8
9   void FlujoAgua()
10  {
11      Instantiate(drop);
12  }
```

Figura 6.16: Función para controlar el flujo de agua

- **Control mediante Leap Motion**

Para implementar la interacción mediante el dispositivo *Leap Motion*, se crea un nuevo script y se añade al escenario el objeto *Leap Rig Base*, que contendrá todos los componentes que permitirán crear el entorno de realidad virtual e interactuar con nuestras manos. Además, se agrega el componente *Interaction Behaviour* a la llave del grifo para permitir la interacción. Se puede consultar más información de dichos componentes en la Sección 6.3.4.

En este nuevo script ya no es necesario provocar un movimiento en la llave mediante una función específica. Se moverá al aplicar una fuerza con nuestras manos. No obstante, tendremos que bloquear la posición del objeto y permitir que solo pueda girar sobre sí mismo, sino sería posible mover la llave en cualquier dirección. Esta configuración se realizará sobre el componente *Rigidbody*, encargado de la física del objeto. Por otra parte, se usará el eje de rotación *z* para comprobar la rotación de la llave, en función de ese valor se identificará cuanto se ha abierto el grifo y generaremos más o menos agua.

En la Figura 6.17, se puede observar la función *FixedUpdate()*, donde comprobaremos continuamente el valor del eje *z*.

```
1   void FixedUpdate()
2   {
3       posAuxY = Mathf.RoundToInt(trans.rotation.eulerAngles.y);
4
5       // Damos ese margen de +5 para evitar llamar continuamente
    a RegularWater, nos interesa un cierto movimiento
6       if (posAuxY > 0 && (posAuxY > posY+5 || posAuxY < posY-5))
7       {
8           posY = posAuxY;
9           RegularWater();
10      }
11  }
```

Figura 6.17: Función para detectar la variación del eje *z*

Por otra parte, en la Figura 6.18, podemos ver la función que se ocupará de modificar el índice del array dependiendo de la posición de la llave.

```
    void RegularWater()
    {
        if (posY < 10)
        {
            indice = 0;
            CancelInvoke("FlujoAgua");
        } else if (posY > 10 && posY <= 50)
        {
            indice = 1;
            GenerateWater();
        }
        else if (posY > 50 && posY <= 100)
        {
            indice = 3;
            GenerateWater();
        }
        else if (posY > 100 && posY <= 160)
        {
            indice = 4;
            GenerateWater();
        }
```

Figura 6.18: Función para modificar el índice del array

Las funciones que controlan y generan el flujo de agua se mantienen igual, se pueden consultar en la figura 6.16.

Finalmente, en la figura 6.19 se puede ver una captura realizada durante la ejecución del escenario.

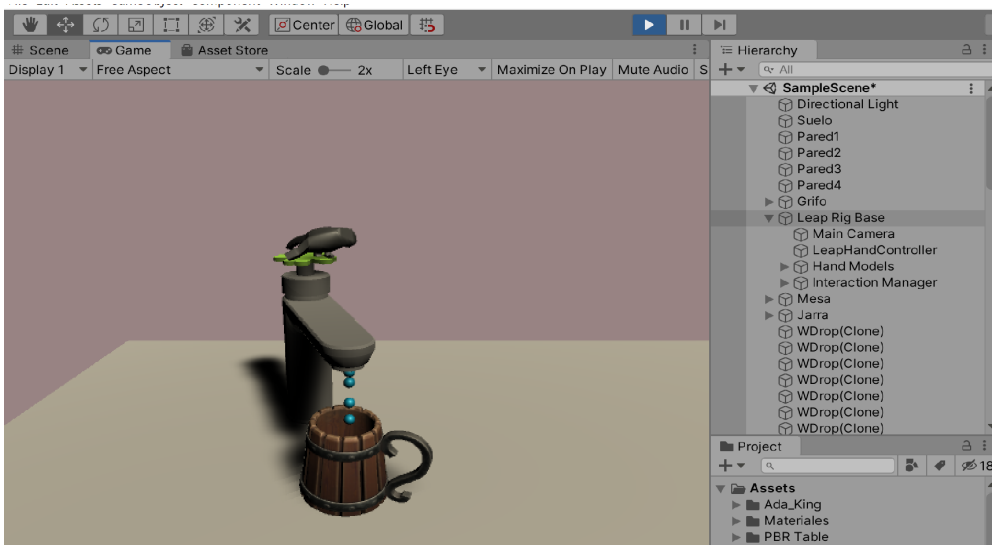

Figura 6.19: Ejecución del escenario *Grifo*

6.3.6 Sprint 5

Fecha: 25/09/2020-22/10/2020

En esta iteración se realizan tres escenarios denominados *Cubiertos*, *Frutero* y *Menu*. Además se incorpora la interfaz de usuario y se configura la exportación de datos de juego.

- **Escenario *Cubiertos***

  En este escenario, el objetivo es agarrar un cuchillo y un tenedor y cortar un filete. Con ello se pretende trabajar los movimientos de mano que implican dicha acción, ya que es una tarea cotidiana que representa una dificultad para ciertas personas con diversidad funcional.

  Para una mayor facilidad en el manejo de los cubiertos, en cuanto se cojan de la mesa, estos quedarán fijados a las manos ya que el sistema *Leap Motion* no es totalmente preciso en el manejo de objetos que no tengan una forma geométrica básica. No obstante, esto no supone ningún inconveniente para cumplir con el objetivo fijado. Para anclar un objeto a una mano se hace uso de un nuevo elemento incluido en los módulos de *Leap Motion*. Se trata del elemento *Attachment Hands* que estará formado a su vez por los elementos *Attachment Hand (Left)* y *Attachment Hand (Right)*.

  Primeramente, en *Attachment Hands* tendremos que seleccionar las zonas de la mano en donde queremos anchar objetos, en este caso únicamente vamos a seleccionar la palma. Podemos ver dicha configuración en la Figura 6.20.

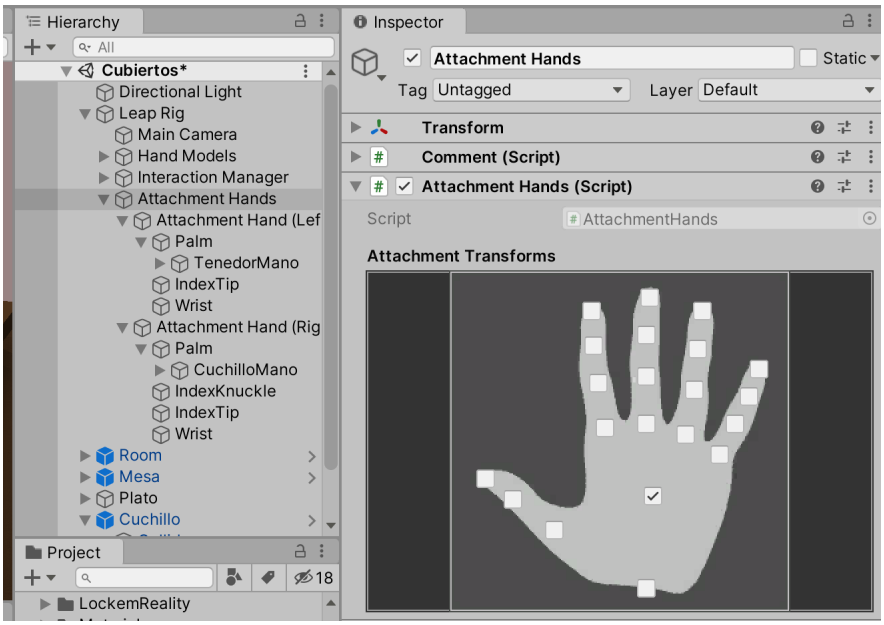

Figura 6.20: Elemento *Attachment Hands*

A continuación, tendremos que añadir el objeto que se desea anclar como hijo de la parte de la mano que hemos activado, en este caso de la palma. Podemos ver en la Figura 6.20 a los objetos *TenedorMano* y *CuchilloMano* como hijos del elemento *palm* de cada mano.

Por otra parte, se creará un script que ancle los cubiertos a las manos una vez que se cogen de la mesa, para dotar de mayor realismo al escenario. En la Figura 6.21 , se puede consultar dicho script.

```
using Leap.Unity.Interaction;
using UnityEngine;

public class ControlCubiertos : MonoBehaviour
{
    private InteractionBehaviour IB;
    public GameObject cubierto;
    void Start()
    {

        IB = GetComponent<InteractionBehaviour>();
        Debug.Log(Data.name);
    }

    void Update()
    {
        //Si agarramos el cubiero de la mesa, desactivamos el mismo
    y activamos el cubierto colocado en la mano
        if (IB.isGrasped)
        {
            gameObject.SetActive(false);
            cubierto.SetActive(true);
        }
    }
}
```

Figura 6.21: Script control cubiertos

Para lograr más realismo, a medida que se realizan cortes en la carne, esta se irá transformando en trozos cada vez más pequeños. Para ello, se han utilizado 4 modelos 3D de carne y dos scripts, uno para detectar el número de cortes en cada trozo y otro para que en función del número de cortes se vaya activando un modelo u otro de carne. En las Figuras 6.22 y 6.23 se puede consultar el código de ambos scripts.

```
using UnityEngine;

public class DetectarCorte : MonoBehaviour
{
    public ControlCorte ControlCorte;
    public GameObject sonidoCorte;

    private void OnTriggerEnter(Collider other)
    {
        // Si el cuchillo toca a la carne, detectamos que ha cortado
        if (other.gameObject.name == "CuchilloMano")
        {
            ControlCorte.Actualizar();

        }
    }
}
```

Figura 6.22: Script detección de corte

```
using UnityEngine;

public class ControlCorte : MonoBehaviour
{
    public GameObject carneEntera;
    public GameObject carneCortada1;
    public GameObject carneCortada2;
    public GameObject carneCortada3;
    private int nCortes;

    private void Start()
    {
        nCortes = 0;
    }

    public void Actualizar()
   {
        if (nCortes < 30)
        {
            nCortes++;

            if (nCortes == 10)
            {
                carneEntera.SetActive(false);
                carneCortada1.SetActive(true);
            }
            else if (nCortes == 20)
            {
                carneCortada1.SetActive(false);
                carneCortada2.SetActive(true);
            }
            else if (nCortes == 30)
            {
                carneCortada2.SetActive(false);
                carneCortada3.SetActive(true);
            }
        }
    }
}
```

Figura 6.23: Script control de corte de carne

Por último, en la Figura 6.24, se puede visualizar el escenario finalizado.

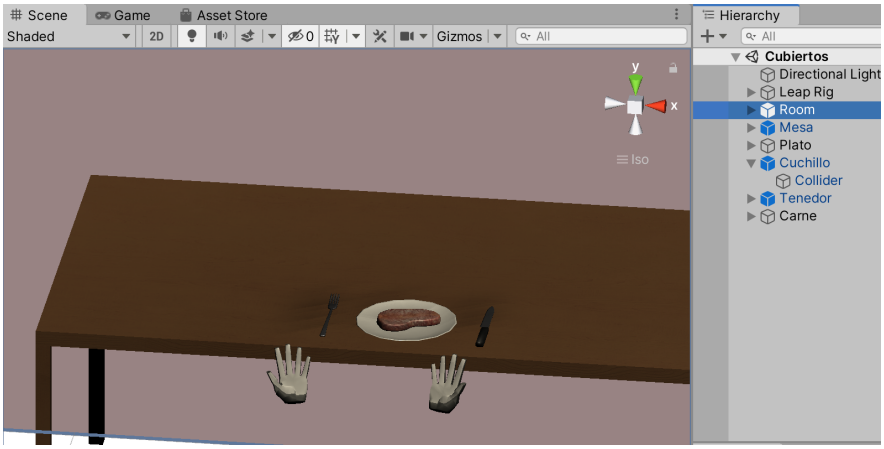

Figura 6.24: Escenario *Cubiertos*

- **Escenario *Frutero***

  En este escenario el usuario deberá coger una serie de piezas de fruta y colocarlas en un frutero. Con ello, se pretende mejorar el agarre y la manipulación de elementos con las manos. Además, se consigue trabajar dicho problema a través de una tarea muy común en la vida cotidiana.

  Para configurar el entorno, se importan los modelos 3D correspondientes a las piezas de fruta, al frutero y a una mesa. A continuación, se añade el elemento *Leap Rig Base* y se agrega el componente *Interaction Behaviour* a cada una de las piezas de fruta, permitiendo así la interacción con dichos objetos.

  En la Figura 6.25, se puede visualizar una captura de la ejecución del escenario.

Figura 6.25: Ejecución del escenario *Frutero*

- **Interfaz de usuario**

Para realizar la interfaz de usuario se emplea la herramienta *UI* de *Unity*. Por una parte, se crea una pantalla de inicio en donde el usuario deberá introducir sus datos. Dicha información se empleará posteriormente como medio de identificación en el registro de actividad. Para configurar la interfaz, se añade un título, dos campos de entrada de texto y un botón. Podemos ver el resultado en la Figura 6.26.

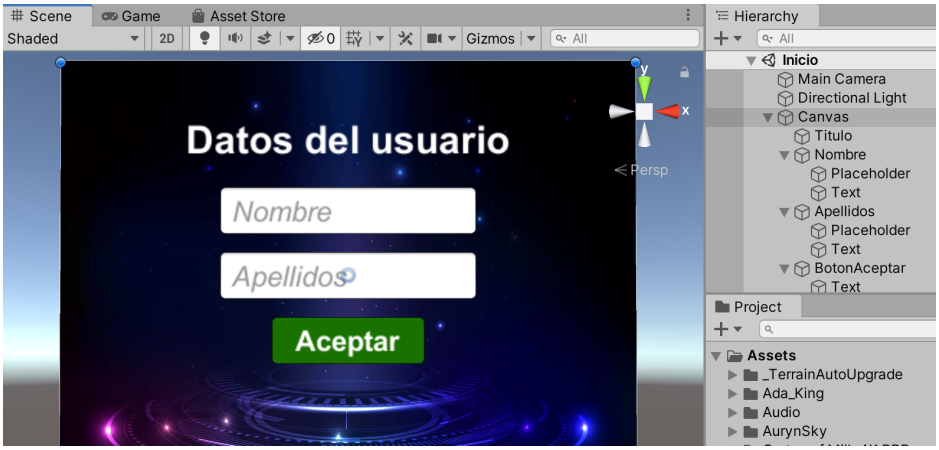

Figura 6.26: Interfaz de usuario

Además, se programan dos scripts, uno para guardar los datos de usuario introducidos en los campos de entrada (Figura 6.28) y otro que nos permitirá cambiar de una escena a otra del juego, así como salir de la aplicación (Figura 6.27).

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ControlEscenas : MonoBehaviour
{
    public void CambiarEscena(string nombreEscena)
    {
        SceneManager.LoadScene(nombreEscena);
    }
    public void Salir()
    {
        Application.Quit();
    }
}
```

Figura 6.27: Script de cambio de escena

```
using UnityEngine;
using UnityEngine.UI;

public class Usuario : MonoBehaviour
{
    static public string nombre = "invitado/a";
    static public string apellidos = "";
    public InputField campoNombre;
    public InputField campoApellidos;

    public void guardarUsuario()
    {
        if(campoNombre.text.Length > 0 &&
    campoNombre.text.Substring(0,1) != " ")
        {
            nombre = campoNombre.text;
        }

        if (campoApellidos.text.Length > 0 &&
    campoApellidos.text.Substring(0, 1) != " ")
        {
            apellidos = campoApellidos.text;
        }
    }
}
```

Figura 6.28: Script de guardado de datos de usuario

Al pulsar el botón *Aceptar* de la pantalla principal se llamará primeramente a la función *GuardarUsuario* y a continuación, a la función *CambiarEscena* que nos llevará al escenario *Menu* . Podemos ver la configuración del botón en la Figura 6.29.

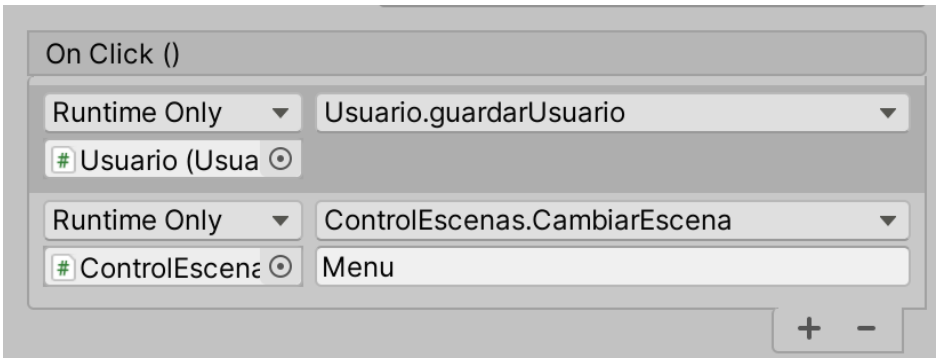

Figura 6.29: Configuración del botón *Aceptar*

- **Escenario *Menú***

  Se trata de un escenario sencillo, pero de gran importancia en la aplicación final. Su finalidad es enlazar todos los escenarios, de modo que el usuario seleccionará que acti-

vidad desea realizar y se producirá la transición a dicho escenario. Así mismo, también permitirá cerrar la aplicación. Para su desarrollo se han creado cuatro botones usando objetos *Cube*, disponibles en *Unity*. Una vez creados, se les ha añadido el componente *Interaction Button* obtenido de los módulos de *Leap Motion*. Este componente es similar al elemento *Interaction Behaviour*, usado en otros escenarios, pero orientado a interactuar con botones. Además, se incorporó el elemento *Leap Rig Base* y se configuró cada botón para cambiar a la escena correspondiente. Para ello se hizo uso del script *ControlEscenas*, creado anteriormente (Figura 6.27).

En la Figura 6.30, se puede ver la configuración de uno de los botones y en la Figura 6.31 una captura del escenario final.

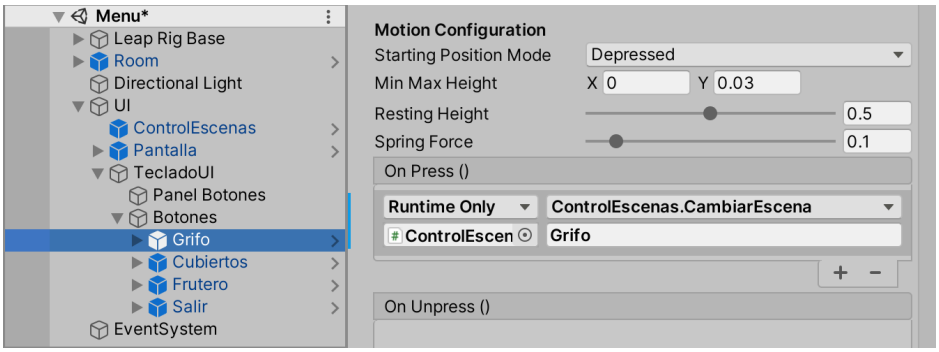

Figura 6.30: Configuración botón *Grifo*

Figura 6.31: Escenario *Menú*

También se añadió un botón *Salir* en cada uno de los entornos de juego que nos llevará al escenario *Menu*. La configuración es la misma que la que se puede observar en la Figura 6.30, pero apuntando al escenario *Menu* en lugar de al escenario *Grifo*. En la Figura 6.32, se puede observar una de las escenas tras incorporar el botón.

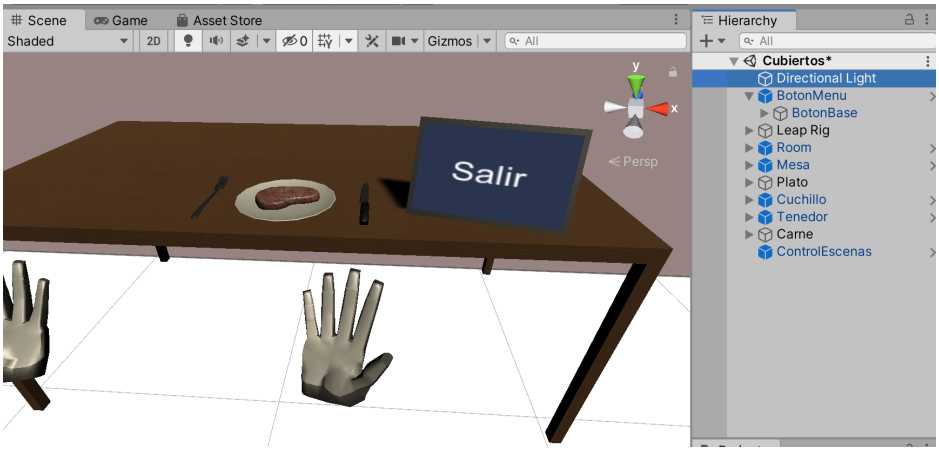

Figura 6.32: Escenario con botón *Salir*

- **Exportación de datos de juego**

Uno de los objetivos del proyecto es la exportación de datos de juego para poder ser evaluados por un terapeuta. En concreto, se guardará el valor máximo, en grados, que se ha conseguido mover cada mano en los ejes *X*, *Y* y *Z*. De modo que se podrá evaluar el movimiento lateral, vertical y rotacional de cada mano.

Para programar esta funcionalidad, se creó un script que mediante una función registra de forma continua el valor de los ejes de posición de nuestras manos. A través de otra función, se realiza una comparación de los valores que se van capturando y guarda los máximos. Por último, otra función exportará los datos a un archivo independiente para cada usuario. En este archivo, además de contener los valores correspondientes al movimiento de cada mano, se registrarán otros datos como el nombre de la actividad, el tiempo de duración o la fecha de realización.

Este script se añadirá a cada entorno de juego, de modo que al iniciar un escenario se empezarán a capturar los datos y una vez pulsemos el botón *Salir*, estos serán exportados.

En la Figura 6.33, podemos observar un ejemplo de archivo generado para un usuario.

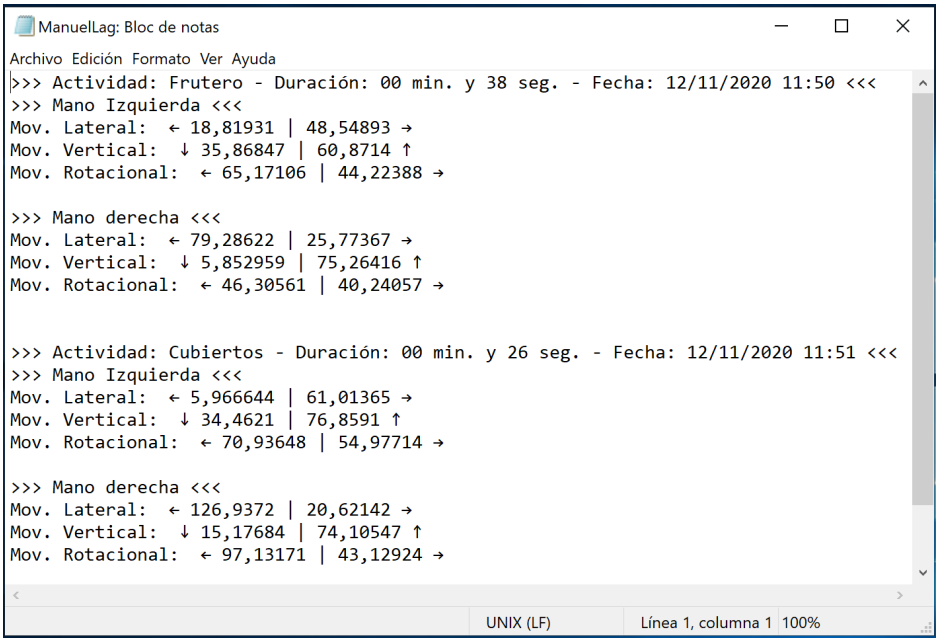

Figura 6.33: Archivo de los datos de juego de un usuario

6.3.7 Sprint 6

Fecha: 22/10/2020-05/11/2020

En este sprint se realiza el escenario denominado *Desayuno*. Se incorporan señales visuales y de audio a las escenas para lograr un mayor realismo y para proporcionar *feedback* al usuario. Además, se integran los escenarios creados en entornos reales.

- **Escenario *Desayuno***

  En este escenario, el usuario deberá seleccionar entre un conjunto de objetos de una mesa, los elementos necesarios para preparar el desayuno y llevarlos a otra mesa. Se pretende trabajar un problema de diversidad funcional cognitiva relacionada con la identificación de elementos, usando para ello una tarea de la vida cotidiana. Además, indirectamente también se trabajará la movilidad de las manos, al igual que en el resto de escenarios.

  Para la realización de este escenario, se importaron una serie de modelos 3D que representan elementos de uso común en un desayuno y otros objetos que nada tienen que ver con dicho contexto. Por otra parte, se añadió el elemento *Leap Rig Base* y se agregó el componente *Interaction Behaviour* a cada objeto de la escena, permitiendo así la manipulación de los mismos.

  En la Figura 6.35, se puede visualizar el escenario con cada uno de los elementos.

Figura 6.34: Escenario *Desayuno*

- **Incorporación de señales visuales y de audio**

  Para dotar de mayor realismo a los diferentes entornos, se ha añadido audio a las diferentes acciones que suceden durante la ejecución de cada entorno. Además, se han añadido una serie de señales visuales y de audio que proporcionan información al usuario sobre su progreso en el logro de los objetivos de cada prueba.

  En el escenario *Grifo*, cuando movamos la llave se reproducirá un audio simulando el sonido que haría en la realidad. También se agregó audio para simular el caudal de agua. Por otra parte, se añadió un panel con tres objetos que representan una estrella. Cada estrella estará asociada a un objetivo del juego, de modo que el logro de cada objetivo iluminará su estrella correspondiente y se emitirá un sonido. Además, cuando se consigan los tres objetivos, se emitirá un sonido especial para indicar el final del juego. Para implementar estas funcionalidades se modificó el script *RegularWater* creado en el Sprint 4 (Sección 6.3.5). Además, se tuvo que crear un *GameObject* para cada clip de audio con el componente *Audio Source* proporcionado por Unity (Figura 6.35) y mediante el cual se importará el archivo de audio.

  En la Figura 6.36, se puede visualizar el escenario tras añadir los elementos indicados.

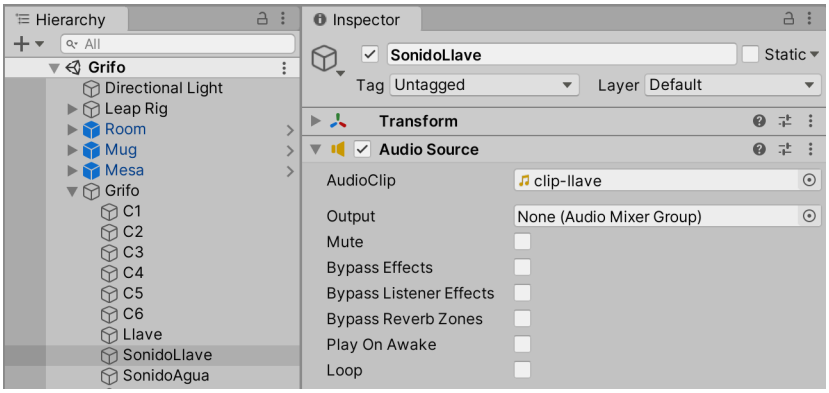

Figura 6.35: Objeto con el componente *Audio Source* de *Unity*

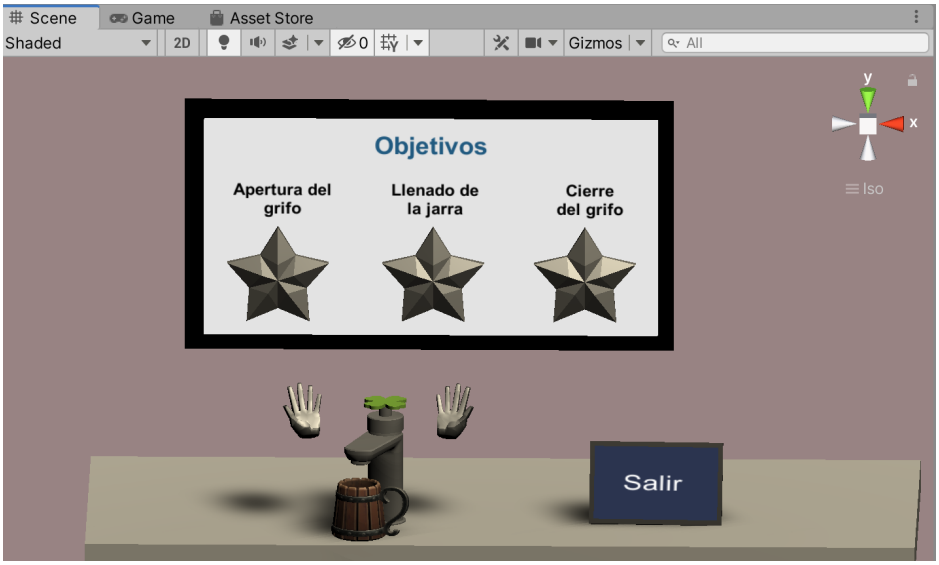

Figura 6.36: Escenario *Grifo* con señales visuales y de audio

En el escenario *Cubiertos*, se reproducirá un sonido simulando el corte de la carne. Por otra parte, el progreso en la actividad se podrá comprobar mediante una barra luminosa y con un indicador de porcentaje, que irán variando a medida que se vayan realizando cortes. Una vez la barra llegue al 100%, se reproducirá un sonido que indicará el final de la actividad. Para implementar esta funcionalidad se modificó el script *ControlCorte* realizado en el Sprint 5 (Sección 6.3.6).

En la Figura 6.37, se puede visualizar una captura del escenario en ejecución.

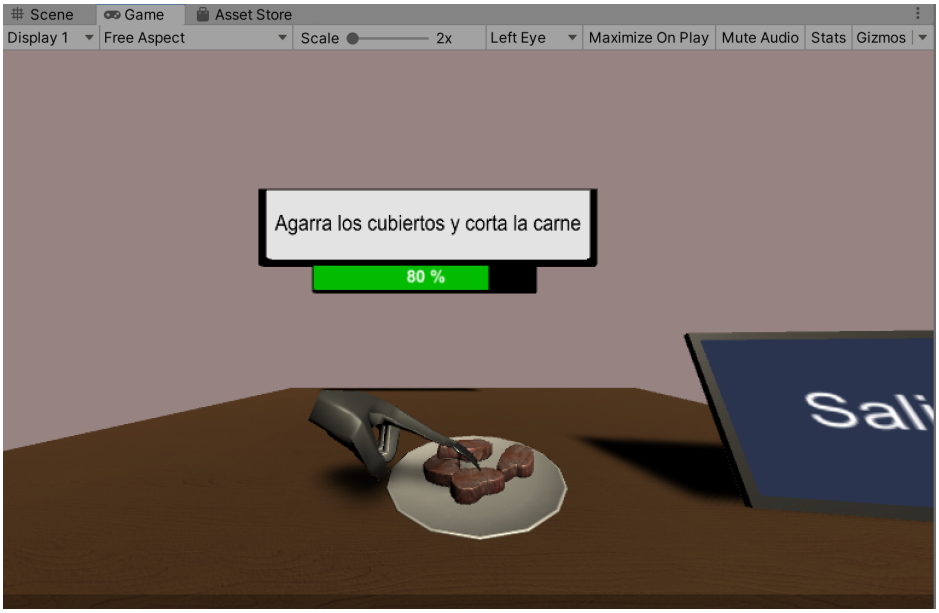

Figura 6.37: Escenario *Cubiertos* con señales visuales y de audio

En el escenario *Frutero*, se reproducirá un audio que simulará el sonido real del contacto con las manos. En cuanto al seguimiento de los objetivos, se hará uso de los modelos 3D de las diferentes piezas de fruta para crear un panel, en donde cada fruta se mostrará en blanco y negro. Cada fruta del panel estará asociada al objeto real que agarraremos con la mano, de modo que cuando metamos una pieza de fruta en el frutero, su representación en el panel pasará de estar en blanco y negro a color. También se emitirá un sonido cada vez que se logre meter una de las frutas en el frutero. Además, una vez se metan todas las frutas, se emitirá un sonido especial que indicará el final del juego.

Para ello, se creó un script que detectará la entrada de cada fruta en el frutero y que controlará la reproducción de los diferentes sonidos y la conversión de blanco y negro a color de cada fruta del panel.

En la Figura 6.38, se puede observar una parte de la función que detectará el tipo de fruta que se ha introducido en el frutero y en la Figura 6.39 la función que emitirá el sonido que indicará que se han cumplido todos los objetivos.

```
1
2     private void OnTriggerEnter(Collider other)
3     {
4         string nombreFruta = other.gameObject.name;
5
6         switch (nombreFruta)
7         {
8             case "Manzana":
9                 if (!frutasOK[0])
10                {
11                    frutasOK[0] = true;
12
    sonidoObjetivo.GetComponent<AudioSource>().Play();
13
    ObjetivosFruta.transform.GetChild(0).gameObject.GetComponent
14                      <MeshRenderer>().material = Manzana;
15                  controlFruta(frutasOK);
16              }
17              break;
18          case "Banana":
19              if (!frutasOK[1])
20              {
21                  frutasOK[1] = true;
22
    sonidoObjetivo.GetComponent<AudioSource>().Play();
23
    ObjetivosFruta.transform.GetChild(1).gameObject.GetComponent
24                      <MeshRenderer>().material = Banana;
25                  controlFruta(frutasOK);
26              }
27              break;
```

Figura 6.38: Parte del código del Script *DetectarFruta*

```
1
2     private void controlFruta(bool[] frutasOk)
3     {
4         bool[] ObjtFrutero = new bool[5] { true, true, true, true,
    true };
5
6         if (frutasOk.SequenceEqual(ObjtFrutero))
7         {
8             sonidoObjetivoFinal.GetComponent<AudioSource>().Play();
9         }
10    }
```

Figura 6.39: Función *ControlFruta* del script *DetectarFruta*

En la Figura 6.40, se puede visualizar una captura del escenario en ejecución.

Figura 6.40: Escenario *Frutero* con señales visuales y de audio

Por último, la implementación del escenario *Desayuno* se realizó de igual modo que el escenario *Frutero*, de modo que cada vez que coloquemos uno de los elementos de desayuno en la mesa indicada, su representación en el panel cambiará de blanco y negro a color, a la vez que se reproducirá un sonido. Se usó la estructura del script *DetectarFruta*, pero adaptando las funciones correspondientes a las Figuras 6.38 y 6.39.

En la Figura 6.41, se puede visualizar una captura del entorno en ejecución.

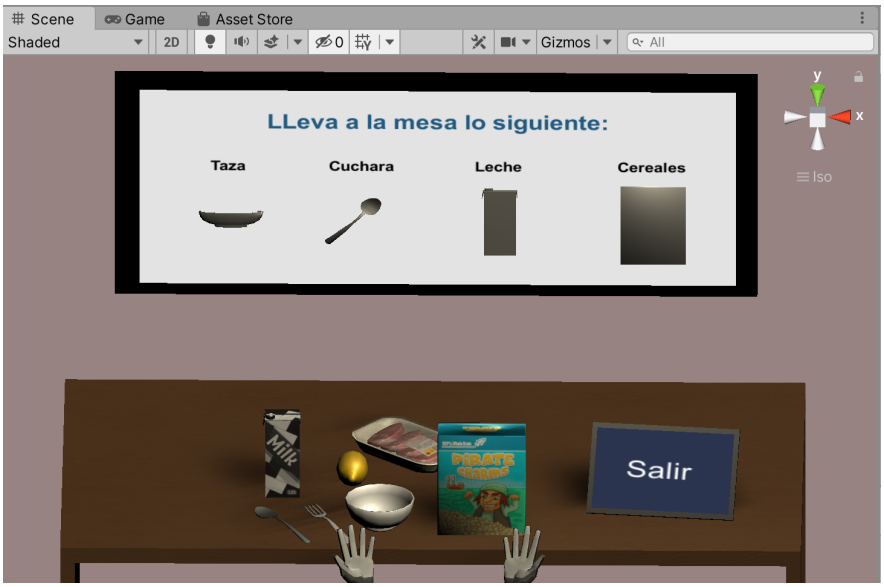

Figura 6.41: Escenario *Desayuno* con señales visuales y de audio

- **Integración de los escenarios en entornos reales**

Para finalizar, se ha obtenido en el *Asset Store* de *Unity* un paquete con las diversas estancias que podemos encontrar en un hogar y se ha procedido a integrar cada escenario desarrollado en la estancia más apropiada. De modo que, los escenarios *Grifo* (Figura 6.42), *Cubiertos* (Figura 6.43), *Frutero* (Figura 6.44)y *Desayuno* (Figura 6.45) se han integrado en una cocina-comedor y el escenario *Menu* (Figura 6.46) en una sala.

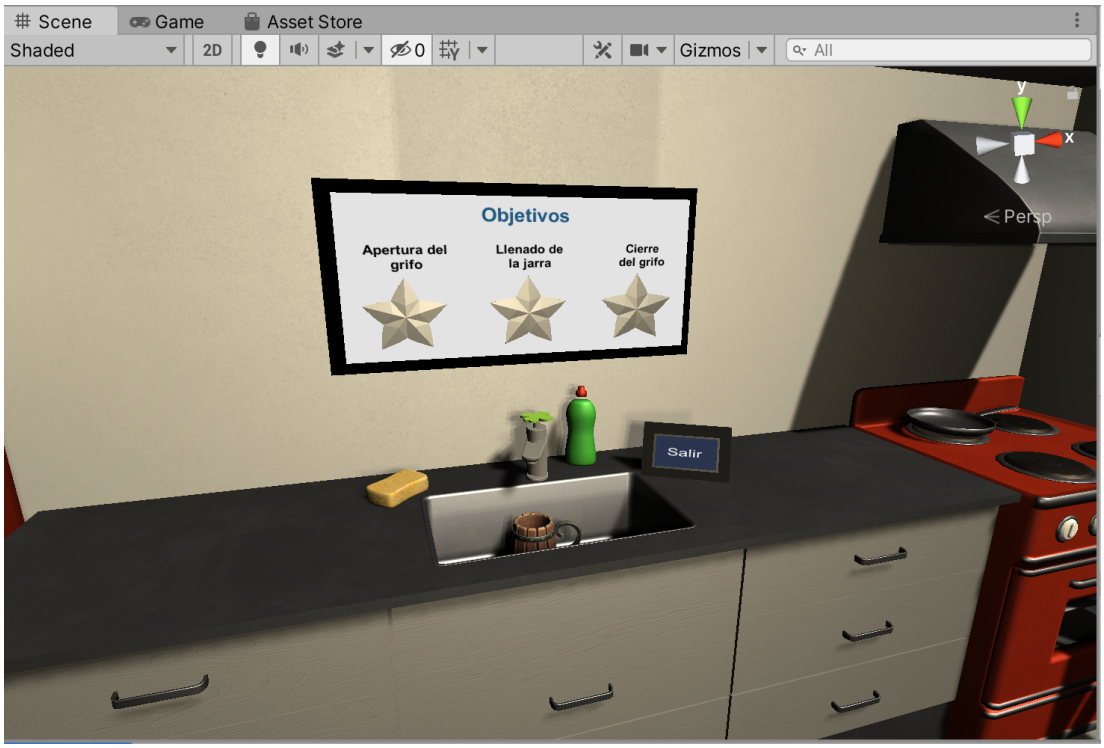

Figura 6.42: Escenario *Grifo* integrado en una cocina-comedor

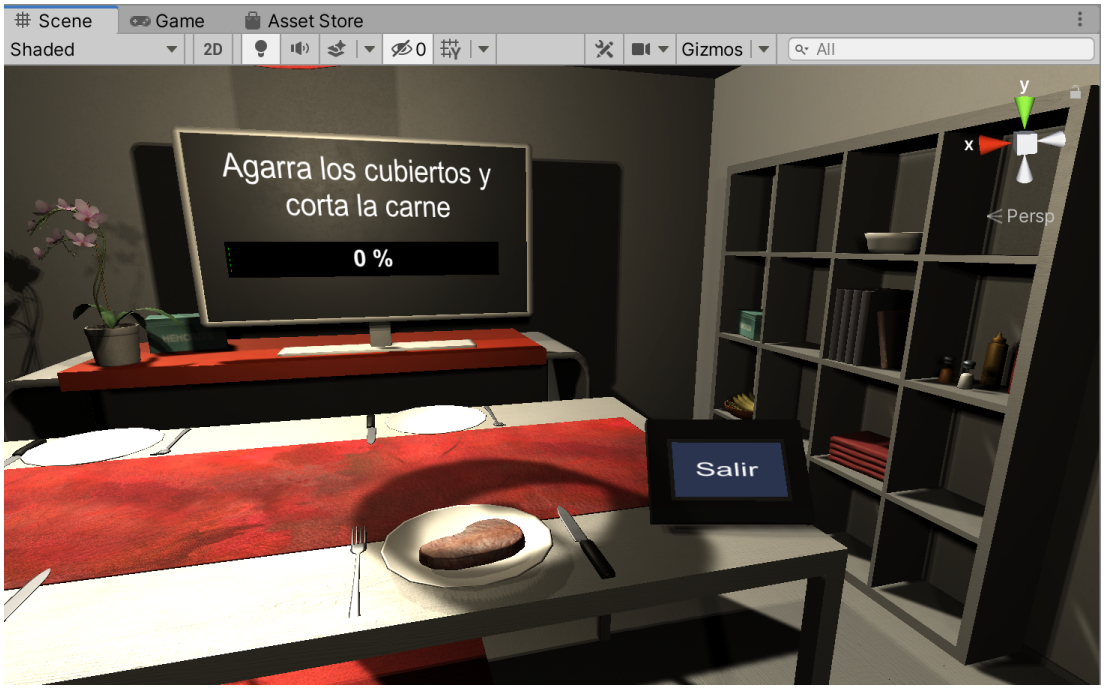

Figura 6.43: Escenario *Cubiertos* integrado en una cocina-comedor

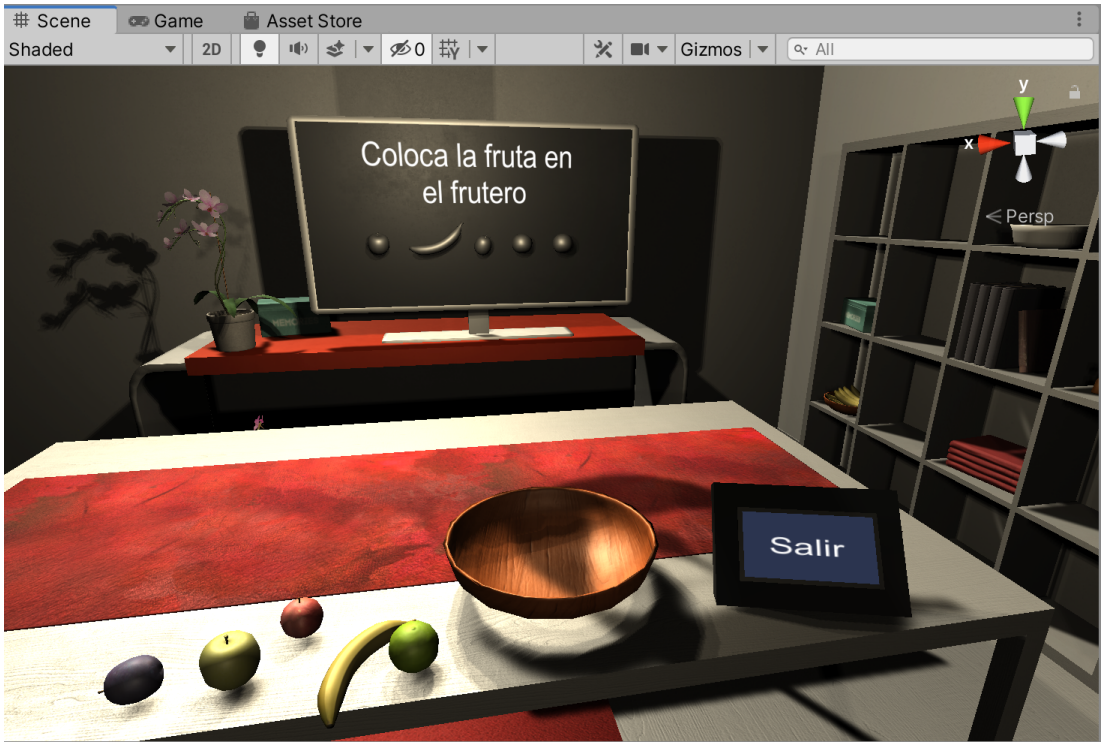

Figura 6.44: Escenario *Frutero* integrado en una cocina-comedor

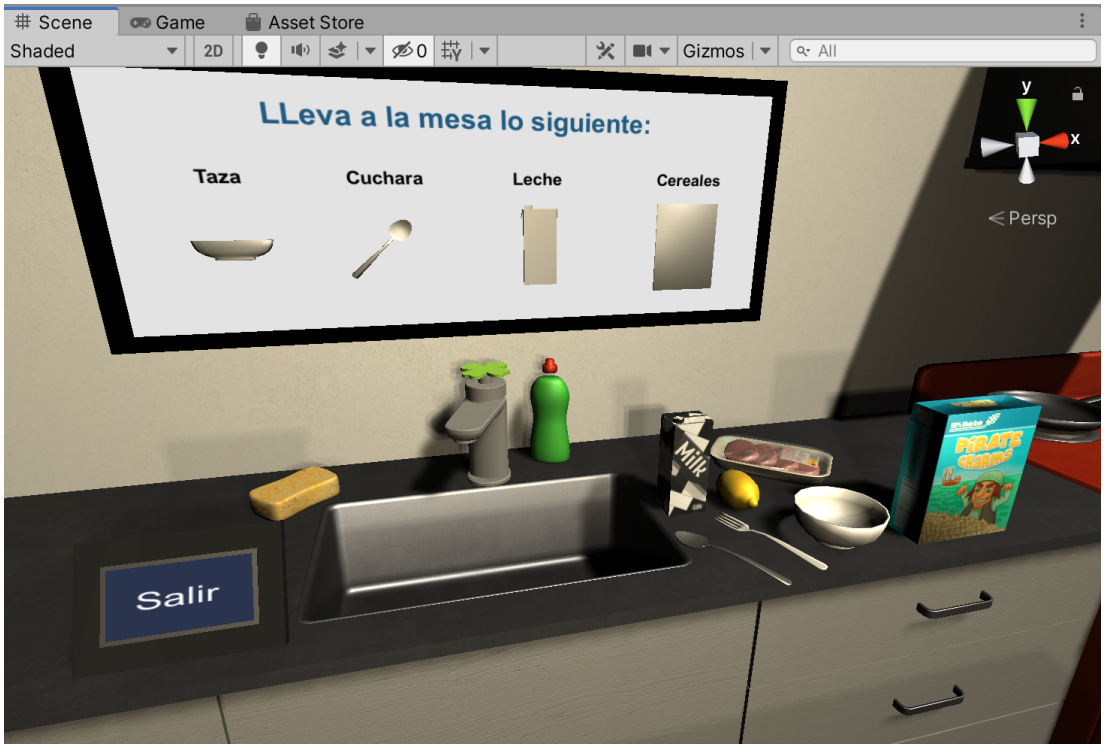

Figura 6.45: Escenario *Desayuno* integrado en un cocina-comedor

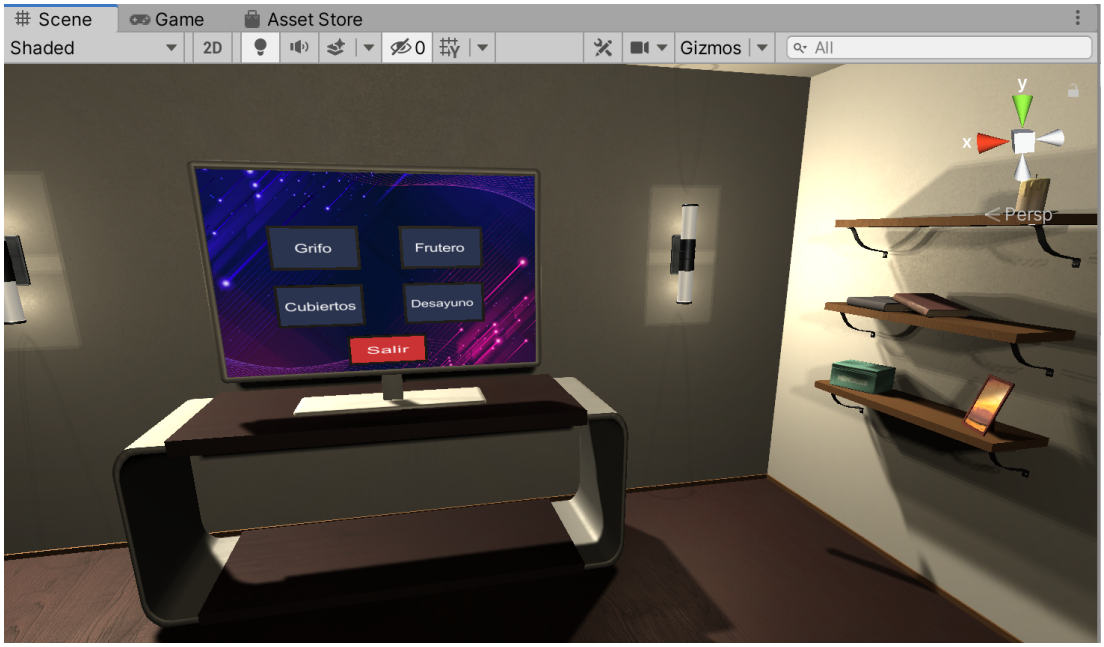

Figura 6.46: Escenario *Menú* integrado en una sala

6.3.8 Sprint 7

Fecha: 05/11/2020-16/11/2020

En esta interacción, por una parte, se realizó la memoria del proyecto. Se empezó por recopilar y ordenar toda la documentación recadada durante el desarrollo del mismo. Posteriormente, se realizó un esquema con los apartados principales en los que se estructuraría la memoria. Por último, se redactó la misma y se realizó la entrega del producto para su aprobación. Como resultado final de la revisión del Sprint se consideró que el trabajo realizado cumplía con los objetivos indicados por el cliente (*Product Owner*) y se aprobó su finalización.

6.4 Pruebas

La realización de pruebas es imprescindible en el desarrollo de un cualquier proyecto software. En cada uno de los sprints se han realizado pruebas unitarias para comprobar el correcto funcionamiento de cada una de las funciones implementadas, en el contexto de *Unity*, esto se correspondería a la comprobación de los diferentes componentes. A continuación, se han realizado pruebas de integración para evaluar la correcta interacción entre los diferentes componentes. Posteriormente, se han realizado las pruebas de sistema, para ello primero se ha comprobado la funcionalidad completa de cada uno de los escenarios y posteriormente se han realizado pruebas de la aplicación en conjunto. Por último, los directores del proyecto han procedido a realizar las pruebas de aceptación, con la finalidad de comprobar que se habían cumplido todos los requerimientos del proyecto. Cabe destacar, la intención de realizar pruebas con usuarios reales. No obstante, no ha sido posible debido a las restricciones de movilidad y seguridad existentes.

Capítulo 7

Conclusiones

Tras finalizar el proyecto, podemos concluir que se han alcanzado los objetivos fijados inicialmente, implementando una herramienta de realidad virtual personalizada para la rehabilitación de personas con diversidad funcional. De modo que se han desarrollado cuatro escenarios que simulan entornos de la vida real y que permiten al usuario realizar una serie de tareas de la vida cotidiana, con la finalidad de trabajar aquellas dificultades de índole física y/o cognitiva.

Resaltar que, también se han podido incorporar satisfactoriamente mejoras que han ido surgiendo durante el desarrollo del mismo, como la incorporación de audio en los entornos, lo que supuso aumentar en gran medida el grado de realismo.

Además del resultado final, es decir, los cuatro entornos creados, también hay que destacar el trabajo realizado en cuanto a análisis e investigación, una parte del proyecto que supuso un gran peso en dedicación. Un trabajo que servirá de base para otros proyectos similares o incluso para una ampliación del presente.

Personalmente, este proyecto me ha permitido conocer más de cerca los sistemas de realidad virtual, una tecnología en la que carecía de experiencia. Además, he podido aprender en profundidad nuevas herramientas como *Unity*, a la vez que pude aplicar los conocimientos adquiridos en la carrera. Por otra parte, me ha permitido tener un mayor conocimiento sobre diferentes terapias a seguir con personas con diversidad funcional física o cognitiva, aparte de descubrir herramientas similares a la desarrollada en este proyecto.

Para mí ha sido una satisfacción el haber realizado este trabajo y espero que pueda servir de ayuda a los diferentes especialistas y personas que lo requieran en sus terapias.

Capítulo 8

Líneas futuras

Aunque se ha considerado que los objetivos establecidos inicialmente con este proyecto se han cumplido ampliamente, nos ha parecido de interés el planteamiento de algunas líneas de trabajo futuro.

- Una tarea que consideramos imprescindible realizar a corto plazo es la prueba de la herramienta con usuarios reales, con la finalidad de obtener información sobre su adaptación a la misma y su utilización por parte del personal especializado. De hecho, se ha llegado a contactar con COGAMI[1], pero a causa de las restricciones de movilidad y seguridad existentes, no se han podido formalizar las pruebas con los usuarios de dicha entidad.

- Una posible ampliación del presente proyecto, podría consistir en permitir regular el grado de dificultad de cada escenario en función de las posibilidades del usuario.

  Si tomamos como ejemplo, el escenario que consistía en seleccionar los elementos para preparar el desayuno, se podría implementar una función que, dependiendo de un valor de dificultad generara elementos con un mayor o menor grado de complejidad a la hora de discernir si se trata o no de un objeto que tiene que ver con la acción de desayunar. Además, también se podrían generar un mayor o menor número de objetos erróneos, aumentando la complejidad en la elección.

- Otra posible mejora, sería el medir un mayor número de parámetros durante la ejecución del juego, con la finalidad de ser facilitados al terapeuta que realiza el seguimiento del usuario. Parámetros como, por ejemplo, el tiempo en conseguir cada uno de los objetivos del escenario de forma independiente o el número de veces que se interactuó con un objeto antes de lograr el objetivo. Por ejemplo, en el escenario de colocar la fruta en el frutero, podríamos medir el número de veces que se intentó agarrar cada pieza de fruta antes de conseguir colocarla en el frutero.

[1] Confederación Galega de Persoas con Discapacidade

Bibliografía

[1] "Sala de terapia ocupacional," (acceso: abril 2020). [En línea]. Disponible en: https://www.geriatricarea.com/2017/08/14/las-salas-de-terapia-ocupacional-contribuyen-a-recuperar-la-autonomia-de-casi-la-mitad-de-los-pacientes

[2] W. Castañares, "Realidad virtual, mímesis y simulación," *CIC. Cuadernos de Información y Comunicación*, vol. 16, pp. 59–81, 2011, (acceso: abril 2020). [En línea]. Disponible en: https://www.redalyc.org/articulo.oa?id=93521629004

[3] P. C. P. y J. M. Cancela Carral y I. Machado de Oliveira y G. Rodríguez-Fuentes, "Realidad virtual inmersiva en personas mayores: estudio de casos (immersive virtual reality in older people: a case study)," *Retos*, no. 39, pp. 1001–1005, 2020, (acceso: junio 2020). [En línea]. Disponible en: https://doi.org/10.47197/retos.v0i39.78195

[4] "Leap Motion," (acceso: marzo 2020). [En línea]. Disponible en: https://www.ultraleap.com/product/leap-motion-controller/

[5] U. P. de Barcelona y ADFO, "Realidad virtual para la rehabilitación de personas que han sufrido un ictus," (acceso: abril 2020). [En línea]. Disponible en: https://cit.upc.edu/es/portfolio-item/rv_rehabilitacion_ictus/

[6] "WalkinVR," (acceso: abril 2020). [En línea]. Disponible en: https://www.walkinvrdriver.com/

[7] "HTC VIVE," (acceso: marzo 2020). [En línea]. Disponible en: https://www.vive.com/eu/product/vive/

[8] "Unity," (acceso: marzo 2020). [En línea]. Disponible en: https://unity.com/es

[9] M. Lidon, *Unity 3D*, 1st ed. Marcombo, 2019.

[10] "C#," (acceso: abril 2020). [En línea]. Disponible en: https://docs.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

[11] S. Putier, *C# 7 y Visual Studio 2017 : Los fundamentos del lenguaje*, 1st ed. ENI, 2018.

[12] "Microsoft visual studio," (acceso: abril 2020). [En línea]. Disponible en: https://visualstudio.microsoft.com/es/

[13] "OpenVR," (acceso: abril 2020). [En línea]. Disponible en: https://github.com/ValveSoftware/openvr

[14] "SteamVR," (acceso: abril 2020). [En línea]. Disponible en: https://partner.steamgames.com/doc/features/steamvr/info

[15] "Steam," (acceso: abril 2020). [En línea]. Disponible en: https://store.steampowered.com/?l=spanish

[16] "Plugin SteamVR," (acceso: abril 2020). [En línea]. Disponible en: https://valvesoftware.github.io/steamvr_unity_plugin/

[17] "XR Interaction Toolkit," (acceso: abril 2020). [En línea]. Disponible en: https://blogs.unity3d.com/es/2019/12/17/xr-interaction-toolkit-preview-package-is-here/

[18] "Controlador leap Motion," (acceso: abril 2020). [En línea]. Disponible en: https://developer.leapmotion.com/sdk-leap-motion-controller/

[19] "Modulos Unity leap Motion," (acceso: abril 2020). [En línea]. Disponible en: https://developer.leapmotion.com/unity/

[20] "Metodologías Ágiles," (acceso: abril 2020). [En línea]. Disponible en: https://www.luisan.net/blog/transformacion-digital/que-son-las-metodologias-agiles

[21] "Manifiesto Ágil," (acceso: abril 2020). [En línea]. Disponible en: https://www.tithink.com/es/2018/10/16/metodologias-agiles-que-son-y-para-que-sirven/

[22] "Scrum," (acceso: abril 2020). [En línea]. Disponible en: https://proyectosagiles.org/que-es-scrum/

[23] J.-P. Subra, *Scrum : un método ágil para sus proyectos*, 2nd ed. DataPro, 2020.

[24] "BOE núm. 251," (acceso: noviembre 2020). [En línea]. Disponible en: https://www.boe.es/eli/es/res/2019/10/07/(8)

[25] "Test HTC VIVE," (acceso: abril 2020). [En línea]. Disponible en: https://www.vive.com/us/ready/

[26] "Dispositivos Oculus," (acceso: abril 2020). [En línea]. Disponible en: https://www.oculus.com/compare/?locale=es_ES

[27] "Valve Index," (acceso: abril 2020). [En línea]. Disponible en: https://store.steampowered.com/valveindex

[28] "Coste HTC VIVE," (acceso: noviembre 2020). [En línea]. Disponible en: https://www.pccomponentes.com/htc-vive-gafas-de-realidad-virtual

[29] "Coste Oculus Rift S," (acceso: noviembre 2020). [En línea]. Disponible en: https://www.oculus.com/rift-s/?locale=es_ES

[30] "Coste Valve Index," (acceso: noviembre 2020). [En línea]. Disponible en: https://store.steampowered.com/sub/354231/

[31] "Coste Oculus GO," (acceso: noviembre 2020). [En línea]. Disponible en: https://www.pccomponentes.com/oculus-go-64gb-gafas-de-realidad-virtual

[32] "Coste Oculus Quest," (acceso: noviembre 2020). [En línea]. Disponible en: https://www.pccomponentes.com/oculus-quest-64gb-gafas-de-realidad-virtual

[33] "Unity Learn," (acceso: junio 2020). [En línea]. Disponible en: https://learn.unity.com/

[34] "Manual Unity," (acceso: junio 2020). [En línea]. Disponible en: https://docs.unity3d.com/Manual/

[35] "XR Plug-in Management," (acceso: junio 2020). [En línea]. Disponible en: https://docs.unity3d.com/Manual/com.unity.xr.management.html

[36] "Documentación Leap Motion," (acceso: agosto 2020). [En línea]. Disponible en: https://leapmotion.github.io/UnityModules/index.html